activemq-cpp-3.0.1

Generated by Doxygen 1.6.1

Sat Apr 17 00:41:34 2010

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Data Structure Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 activemq Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **cmsutil**
- namespace **commands**
- namespace **core**
- namespace **exceptions**
- namespace **io**
- namespace **library**
- namespace **state**
- namespace **threads**
- namespace **transport**
- namespace **util**
- namespace **wireformat**

5.1.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

`http://www.apache.org/licenses/LICENSE-2.0`

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.2 activemq::cmsutil Namespace Reference

Data Structures

- class **CachedConsumer**

  A cached message consumer contained within a pooled session.

- class **CachedProducer**

  A cached message producer contained within a pooled session.

- class **CmsAccessor**

  Base class for ***activemq.cmsutil.CmsTemplate*** *(p. 858) and other CMS-accessing gateway helpers, defining common properties such as the CMS* ***cms.ConnectionFactory*** *(p. 978) to operate on.*

- class **CmsDestinationAccessor**

  Extends the `CmsAccessor` *(p. 843) to add support for resolving destination names.*

- class **CmsTemplate**

  `CmsTemplate` *(p. 858) simplifies performing synchronous CMS operations.*

- class **DestinationResolver**

  Resolves a CMS destination name to a `Destination`*.*

- class **DynamicDestinationResolver**

  Resolves a CMS destination name to a `Destination`*.*

- class **MessageCreator**

  Creates the user-defined message to be sent by the `CmsTemplate` *(p. 858).*

- class **PooledSession**

  A pooled session object that wraps around a delegate session.

- class **ProducerCallback**

  Callback for sending a message to a CMS destination.

- class **ResourceLifecycleManager**

  Manages the lifecycle of a set of CMS resources.

- class **SessionCallback**

  Callback for executing any number of operations on a provided CMS Session.

- class **SessionPool**

  A pool of CMS sessions from the same connection and with the same acknowledge mode.

5.3 activemq::commands Namespace Reference

Data Structures

- class **ActiveMQBlobMessage**
- class **ActiveMQBytesMessage**
- class **ActiveMQDestination**
- class **ActiveMQMapMessage**
- class **ActiveMQMessage**
- class **ActiveMQMessageTemplate**
- class **ActiveMQObjectMessage**
- class **ActiveMQQueue**
- class **ActiveMQStreamMessage**
- class **ActiveMQTempDestination**
- class **ActiveMQTempQueue**
- class **ActiveMQTempTopic**
- class **ActiveMQTextMessage**
- class **ActiveMQTopic**
- class **BaseCommand**
- class **BaseDataStructure**
- class **BooleanExpression**
- class **BrokerError**

  This class represents an Exception sent from the Broker.

- class **BrokerId**
- class **BrokerInfo**
- class **Command**
- class **ConnectionControl**
- class **ConnectionError**
- class **ConnectionId**
- class **ConnectionInfo**
- class **ConsumerControl**
- class **ConsumerId**
- class **ConsumerInfo**
- class **ControlCommand**
- class **DataArrayResponse**
- class **DataResponse**
- class **DataStructure**
- class **DestinationInfo**
- class **DiscoveryEvent**
- class **ExceptionResponse**
- class **FlushCommand**
- class **IntegerResponse**
- class **JournalQueueAck**
- class **JournalTopicAck**
- class **JournalTrace**
- class **JournalTransaction**
- class **KeepAliveInfo**
- class **LastPartialCommand**
- class **LocalTransactionId**

- class **Message**
- class **MessageAck**
- class **MessageDispatch**
- class **MessageDispatchNotification**
- class **MessageId**
- class **MessagePull**
- class **NetworkBridgeFilter**
- class **PartialCommand**
- class **ProducerAck**
- class **ProducerId**
- class **ProducerInfo**
- class **RemoveInfo**
- class **RemoveSubscriptionInfo**
- class **ReplayCommand**
- class **Response**
- class **SessionId**
- class **SessionInfo**
- class **ShutdownInfo**
- class **SubscriptionInfo**
- class **TransactionId**
- class **TransactionInfo**
- class **WireFormatInfo**
- class **XATransactionId**

5.4 activemq::core Namespace Reference

Data Structures

- class **ActiveMQAckHandler**

  Interface class that is used to give CMS Messages an interface to Ack themselves with.

- class **ActiveMQConnection**

  Concrete connection used for all connectors to the ActiveMQ broker.

- class **ActiveMQConnectionFactory**
- class **ActiveMQConnectionMetaData**

  This class houses all the various settings and information that is used by an instance of an ***ActiveMQConnection*** *(p. 190) class.*

- class **ActiveMQConnectionSupport**
- class **ActiveMQConstants**

  Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

- class **ActiveMQConsumer**
- class **ActiveMQProducer**
- class **ActiveMQSession**
- class **ActiveMQSessionExecutor**

  Delegate dispatcher for a single session.

- class **ActiveMQTransactionContext**

  Transaction Management class, hold messages that are to be redelivered upon a request to rollback.

- class **DispatchData**

  Simple POCO that contains the information necessary to route a message to a specified consumer.

- class **Dispatcher**

  Interface for an object responsible for dispatching messages to consumers.

- class **MessageDispatchChannel**
- class **Synchronization**

  Transacted Object ***Synchronization*** *(p. 2516), used to sync the events of a Transaction with the items in the Transaction.*

5.5 activemq::exceptions Namespace Reference

Data Structures

- class **ActiveMQException**
- class **BrokerException**

5.6 activemq::io Namespace Reference

Data Structures

- class **LoggingInputStream**
- class **LoggingOutputStream**

  OutputStream filter that just logs the data being written.

5.7 activemq::library Namespace Reference

Data Structures

- class **ActiveMQCPP**

5.8 activemq::state Namespace Reference

Data Structures

- class **CommandVisitor**

  Interface for an Object that can visit the various Command Objects that are sent from and to this client.

- class **CommandVisitorAdapter**

  Default Implementation of a ***CommandVisitor*** *(p. 885) that returns NULL for all calls.*

- class **ConnectionState**
- class **ConnectionStateTracker**
- class **ConsumerState**
- class **ProducerState**
- class **SessionState**
- class **Tracked**
- class **TransactionState**

5.9 activemq::threads Namespace Reference

Data Structures

- class **CompositeTask**

  Represents a single task that can be part of a set of Tasks that are contained in a ***CompositeTaskRunner*** *(p. 908).*

- class **CompositeTaskRunner**

  A ***Task*** *(p. 2520) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

- class **DedicatedTaskRunner**
- class **Task**

  Represents a unit of work that requires one or more iterations to complete.

- class **TaskRunner**

5.10 activemq::transport Namespace Reference

Namespaces

- namespace **correlator**
- namespace **failover**
- namespace **logging**
- namespace **mock**
- namespace **tcp**

Data Structures

- class **AbstractTransportFactory**

  *Abstract implementation of the **TransportFactory** (p. 2614) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2614) instances.*

- class **CompositeTransport**

  *A Composite **Transport** (p. 2608) is a **Transport** (p. 2608) implementation that is composed of several Transports.*

- class **DefaultTransportListener**
- class **IOTransport**

  *Implementation of the **Transport** (p. 2608) interface that performs marshaling of **commands** (p. 59) to IO streams.*

- class **Transport**

  *Interface for a **transport** (p. 67) layer for command objects.*

- class **TransportFactory**

  Defines the interface for Factories that create Transports or TransportFilters.

- class **TransportFilter**

  *A filter on the **transport** (p. 67) layer.*

- class **TransportListener**

  *A listener of asynchronous **exceptions** (p. 62) from a command **transport** (p. 67) object.*

- class **TransportRegistry**

  *Registry of all **Transport** (p. 2608) Factories that are available to the client at runtime.*

5.11 activemq::transport::correlator Namespace Reference

Data Structures

- class **FutureResponse**

  A container that holds a response object.

- class **ResponseCorrelator**

  *This type of **transport** (p. 67) filter is responsible for correlating asynchronous responses with requests.*

5.12 activemq::transport::failover Namespace Reference

Data Structures

- class **BackupTransport**
- class **BackupTransportPool**
- class **CloseTransportsTask**
- class **FailoverTransport**
- class **FailoverTransportFactory**

  *Creates an instance of a **FailoverTransport** (p. 1296).*

- class **FailoverTransportListener**

  *Utility class used by the **Transport** (p. 2608) to perform the work of responding to events from the active **Transport** (p. 2608).*

- class **URIPool**

5.13 activemq::transport::logging Namespace Reference

Data Structures

- class **LoggingTransport**

  *A **transport** (p. 67) filter that logs **commands** (p. 59) as they are sent/received.*

5.14 activemq::transport::mock Namespace Reference

Data Structures

- class **InternalCommandListener**

  Listens for Commands sent from the ***MockTransport*** (p. *1910*).

- class **MockTransport**

  The ***MockTransport*** (p. *1910*) *defines a base level* ***Transport*** (p. *2608*) *class that is intended to be used in place of an a regular protocol* ***Transport*** (p. *2608*) *suck as TCP.*

- class **MockTransportFactory**

  Manufactures MockTransports, which are objects that read from input streams and write to output streams.

- class **ResponseBuilder**

  Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

5.15 activemq::transport::tcp Namespace Reference

Data Structures

- class **TcpTransport**

  *Implements a TCP/IP based **transport** (p. 67) filter, this **transport** (p. 67) is meant to wrap an instance of an **IOTransport** (p. 1480).*

- class **TcpTransportFactory**

  *Factory Responsible for creating the **TcpTransport** (p. 2532).*

5.16 activemq::util Namespace Reference

Data Structures

- class **ActiveMQProperties**

  Implementation of the CMSProperties interface that delegates to a ***decaf::util::Properties*** *(p. 2140) object.*

- class **CompositeData**

  Represents a Composite URI.

- class **LongSequenceGenerator**

  This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

- class **MemoryUsage**
- class **PrimitiveList**

  List of primitives.

- class **PrimitiveMap**

  Map of named primitives.

- class **PrimitiveValueConverter**

  Class controls the conversion of data contained in a ***PrimitiveValueNode*** *(p. 2070) from one type to another.*

- class **PrimitiveValueNode**

  Class that wraps around a single value of one of the many types.

- class **URISupport**
- class **Usage**

5.17 activemq::wireformat Namespace Reference

Namespaces

- namespace **openwire**
- namespace **stomp**

Data Structures

- class **MarshalAware**
- class **WireFormat**

  *Provides a mechanism to marshal **commands** (p. 59) into and out of packets or into and out of streams, Channels and Datagrams.*

- class **WireFormatFactory**

  *The **WireFormatFactory** (p. 2697) is the interface that all **WireFormatFactory** (p. 2697) classes must extend.*

- class **WireFormatNegotiator**

  *Defines a **WireFormatNegotiator** (p. 2721) which allows a **WireFormat** (p. 2693) to.*

- class **WireFormatRegistry**

  *Registry of all **WireFormat** (p. 2693) Factories that are available to the client at runtime.*

5.18 activemq::wireformat::openwire Namespace Reference

Namespaces

- namespace **marshal**
- namespace **utils**

Data Structures

- class **OpenWireFormat**
- class **OpenWireFormatFactory**
- class **OpenWireFormatNegotiator**
- class **OpenWireResponseBuilder**

5.19 activemq::wireformat::openwire::marshal Namespace Reference

Namespaces

- namespace **v1**
- namespace **v2**
- namespace **v3**

Data Structures

- class **BaseDataStreamMarshaller**

  *Base class for all Marshallers that **marshal** (p. 76) DataStructures to and from the wire using the OpenWire protocol.*

- class **DataStreamMarshaller**

  *Base class for all classes that **marshal** (p. 76) **commands** (p. 59) for Openwire.*

- class **PrimitiveTypesMarshaller**

  *This class wraps the functionality needed to **marshal** (p. 76) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

5.20 activemq::wireformat::openwire::marshal::v1 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 155).*
- class **ActiveMQBytesMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).*
- class **ActiveMQDestinationMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 244).*
- class **ActiveMQMapMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 271).*
- class **ActiveMQMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 286).*
- class **ActiveMQObjectMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 317).*
- class **ActiveMQQueueMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 345).*
- class **ActiveMQStreamMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 391).*
- class **ActiveMQTempDestinationMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 407).*
- class **ActiveMQTempQueueMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 424).*
- class **ActiveMQTempTopicMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 441).*
- class **ActiveMQTextMessageMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 457).*
- class **ActiveMQTopicMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 473).*
- class **BaseCommandMarshaller**
  *Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 521).*
- class **BrokerIdMarshaller**

Marshaling code for Open Wire Format for ***BrokerIdMarshaller*** *(p. 599).*

- class **BrokerInfoMarshaller**
  Marshaling code for Open Wire Format for ***BrokerInfoMarshaller*** *(p. 619).*
- class **ConnectionControlMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(p. 954).*
- class **ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 970).*
- class **ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 989).*
- class **ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(p. 1007).*
- class **ConsumerControlMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1037).*
- class **ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(p. 1054).*
- class **ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1074).*
- class **ControlCommandMarshaller**
  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(p. 1087).*
- class **DataArrayResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(p. 1108).*
- class **DataResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(p. 1139).*
- class **DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** *(p. 1203).*
- class **DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1221).*
- class **ExceptionResponseMarshaller**
  Marshaling code for Open Wire Format for ***ExceptionResponseMarshaller*** *(p. 1283).*
- class **FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1360).*
- class **IntegerResponseMarshaller**
  Marshaling code for Open Wire Format for ***IntegerResponseMarshaller*** *(p. 1449).*

- class **JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(p. 1499).*

- class **JournalTopicAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1520).*

- class **JournalTraceMarshaller**

  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1535).*

- class **JournalTransactionMarshaller**

  Marshaling code for Open Wire Format for ***JournalTransactionMarshaller*** *(p. 1551).*

- class **KeepAliveInfoMarshaller**

  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(p. 1566).*

- class **LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1582).*

- class **LocalTransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***LocalTransactionIdMarshaller*** *(p. 1612).*

- class **MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(p. 1791).*

- class **MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(p. 1819).*

- class **MessageDispatchNotificationMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchNotificationMarshaller*** *(p. 1833).*

- class **MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(p. 1848).*

- class **MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(p. 1871).*

- class **MessagePullMarshaller**

  Marshaling code for Open Wire Format for ***MessagePullMarshaller*** *(p. 1906).*

- class **NetworkBridgeFilterMarshaller**

  Marshaling code for Open Wire Format for ***NetworkBridgeFilterMarshaller*** *(p. 1935).*

- class **PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(p. 2007).*

- class **ProducerAckMarshaller**

Marshaling code for Open Wire Format for ***ProducerAckMarshaller*** *(p. 2094).*

- class **ProducerIdMarshaller**

  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(p. 2118).*

- class **ProducerInfoMarshaller**

  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(p. 2135).*

- class **RemoveInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2185).*

- class **RemoveSubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveSubscriptionInfoMarshaller*** *(p. 2206).*

- class **ReplayCommandMarshaller**

  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2222).*

- class **ResponseMarshaller**

  Marshaling code for Open Wire Format for ***ResponseMarshaller*** *(p. 2251).*

- class **SessionIdMarshaller**

  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(p. 2288).*

- class **SessionInfoMarshaller**

  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(p. 2304).*

- class **ShutdownInfoMarshaller**

  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(p. 2356).*

- class **SubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2496).*

- class **TransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(p. 2577).*

- class **TransactionInfoMarshaller**

  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2602).*

- class **WireFormatInfoMarshaller**

  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(p. 2717).*

- class **XATransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***XATransactionIdMarshaller*** *(p. 2736).*

5.21 activemq::wireformat::openwire::marshal::v2 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQBlobMessageMarshaller*** *(p. 159).*
- class **ActiveMQBytesMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQBytesMessageMarshaller*** *(p. 186).*
- class **ActiveMQDestinationMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQDestinationMarshaller*** *(p. 248).*
- class **ActiveMQMapMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQMapMessageMarshaller*** *(p. 275).*
- class **ActiveMQMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQMessageMarshaller*** *(p. 290).*
- class **ActiveMQObjectMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQObjectMessageMarshaller*** *(p. 321).*
- class **ActiveMQQueueMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQQueueMarshaller*** *(p. 349).*
- class **ActiveMQStreamMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQStreamMessageMarshaller*** *(p. 395).*
- class **ActiveMQTempDestinationMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempDestinationMarshaller*** *(p. 411).*
- class **ActiveMQTempQueueMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempQueueMarshaller*** *(p. 428).*
- class **ActiveMQTempTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempTopicMarshaller*** *(p. 445).*
- class **ActiveMQTextMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTextMessageMarshaller*** *(p. 461).*
- class **ActiveMQTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTopicMarshaller*** *(p. 477).*
- class **BaseCommandMarshaller**
  Marshaling code for Open Wire Format for ***BaseCommandMarshaller*** *(p. 528).*
- class **BrokerIdMarshaller**

Marshaling code for Open Wire Format for ***BrokerIdMarshaller*** *(p. 603).*

- class **BrokerInfoMarshaller**
  Marshaling code for Open Wire Format for ***BrokerInfoMarshaller*** *(p. 623).*
- class **ConnectionControlMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(p. 958).*
- class **ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 974).*
- class **ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 993).*
- class **ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(p. 1011).*
- class **ConsumerControlMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1041).*
- class **ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(p. 1058).*
- class **ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1078).*
- class **ControlCommandMarshaller**
  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(p. 1095).*
- class **DataArrayResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(p. 1112).*
- class **DataResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(p. 1143).*
- class **DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** *(p. 1207).*
- class **DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1225).*
- class **ExceptionResponseMarshaller**
  Marshaling code for Open Wire Format for ***ExceptionResponseMarshaller*** *(p. 1287).*
- class **FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1368).*
- class **IntegerResponseMarshaller**
  Marshaling code for Open Wire Format for ***IntegerResponseMarshaller*** *(p. 1445).*

- class **JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(p. 1495).*

- class **JournalTopicAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1512).*

- class **JournalTraceMarshaller**

  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1527).*

- class **JournalTransactionMarshaller**

  Marshaling code for Open Wire Format for ***JournalTransactionMarshaller*** *(p. 1543).*

- class **KeepAliveInfoMarshaller**

  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(p. 1562).*

- class **LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1586).*

- class **LocalTransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***LocalTransactionIdMarshaller*** *(p. 1604).*

- class **MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(p. 1783).*

- class **MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(p. 1811).*

- class **MessageDispatchNotificationMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchNotificationMarshaller*** *(p. 1829).*

- class **MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(p. 1856).*

- class **MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(p. 1861).*

- class **MessagePullMarshaller**

  Marshaling code for Open Wire Format for ***MessagePullMarshaller*** *(p. 1898).*

- class **NetworkBridgeFilterMarshaller**

  Marshaling code for Open Wire Format for ***NetworkBridgeFilterMarshaller*** *(p. 1931).*

- class **PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(p. 1999).*

- class **ProducerAckMarshaller**

Marshaling code for Open Wire Format for ***ProducerAckMarshaller*** *(p. 2098).*

- class **ProducerIdMarshaller**

  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(p. 2110).*

- class **ProducerInfoMarshaller**

  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(p. 2127).*

- class **RemoveInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2181).*

- class **RemoveSubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveSubscriptionInfoMarshaller*** *(p. 2198).*

- class **ReplayCommandMarshaller**

  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2214).*

- class **ResponseMarshaller**

  Marshaling code for Open Wire Format for ***ResponseMarshaller*** *(p. 2241).*

- class **SessionIdMarshaller**

  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(p. 2296).*

- class **SessionInfoMarshaller**

  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(p. 2308).*

- class **ShutdownInfoMarshaller**

  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(p. 2352).*

- class **SubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2504).*

- class **TransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(p. 2585).*

- class **TransactionInfoMarshaller**

  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2594).*

- class **WireFormatInfoMarshaller**

  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(p. 2713).*

- class **XATransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***XATransactionIdMarshaller*** *(p. 2744).*

5.22 activemq::wireformat::openwire::marshal::v3 Namespace Reference

Data Structures

- class **ActiveMQBlobMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQBlobMessageMarshaller*** *(p. 151).*
- class **ActiveMQBytesMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQBytesMessageMarshaller*** *(p. 178).*
- class **ActiveMQDestinationMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQDestinationMarshaller*** *(p. 240).*
- class **ActiveMQMapMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQMapMessageMarshaller*** *(p. 267).*
- class **ActiveMQMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQMessageMarshaller*** *(p. 282).*
- class **ActiveMQObjectMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQObjectMessageMarshaller*** *(p. 313).*
- class **ActiveMQQueueMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQQueueMarshaller*** *(p. 341).*
- class **ActiveMQStreamMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQStreamMessageMarshaller*** *(p. 387).*
- class **ActiveMQTempDestinationMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempDestinationMarshaller*** *(p. 403).*
- class **ActiveMQTempQueueMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempQueueMarshaller*** *(p. 420).*
- class **ActiveMQTempTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTempTopicMarshaller*** *(p. 437).*
- class **ActiveMQTextMessageMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTextMessageMarshaller*** *(p. 453).*
- class **ActiveMQTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTopicMarshaller*** *(p. 469).*
- class **BaseCommandMarshaller**
  Marshaling code for Open Wire Format for ***BaseCommandMarshaller*** *(p. 514).*
- class **BrokerIdMarshaller**

Marshaling code for Open Wire Format for ***BrokerIdMarshaller*** *(p. 595).*

- class **BrokerInfoMarshaller**
  Marshaling code for Open Wire Format for ***BrokerInfoMarshaller*** *(p. 615).*
- class **ConnectionControlMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(p. 950).*
- class **ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 966).*
- class **ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 985).*
- class **ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(p. 1003).*
- class **ConsumerControlMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1033).*
- class **ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(p. 1050).*
- class **ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1070).*
- class **ControlCommandMarshaller**
  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(p. 1091).*
- class **DataArrayResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(p. 1104).*
- class **DataResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(p. 1135).*
- class **DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** *(p. 1199).*
- class **DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1217).*
- class **ExceptionResponseMarshaller**
  Marshaling code for Open Wire Format for ***ExceptionResponseMarshaller*** *(p. 1279).*
- class **FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1364).*
- class **IntegerResponseMarshaller**
  Marshaling code for Open Wire Format for ***IntegerResponseMarshaller*** *(p. 1453).*

- class **JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(p. 1503).*

- class **JournalTopicAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1516).*

- class **JournalTraceMarshaller**

  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1531).*

- class **JournalTransactionMarshaller**

  Marshaling code for Open Wire Format for ***JournalTransactionMarshaller*** *(p. 1547).*

- class **KeepAliveInfoMarshaller**

  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(p. 1558).*

- class **LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1578).*

- class **LocalTransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***LocalTransactionIdMarshaller*** *(p. 1608).*

- class **MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

- class **MessageAckMarshaller**

  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(p. 1787).*

- class **MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(p. 1815).*

- class **MessageDispatchNotificationMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchNotificationMarshaller*** *(p. 1837).*

- class **MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(p. 1852).*

- class **MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(p. 1866).*

- class **MessagePullMarshaller**

  Marshaling code for Open Wire Format for ***MessagePullMarshaller*** *(p. 1902).*

- class **NetworkBridgeFilterMarshaller**

  Marshaling code for Open Wire Format for ***NetworkBridgeFilterMarshaller*** *(p. 1927).*

- class **PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(p. 2003).*

- class **ProducerAckMarshaller**

Marshaling code for Open Wire Format for ***ProducerAckMarshaller*** *(p. 2090).*

- class **ProducerIdMarshaller**
  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(p. 2114).*
- class **ProducerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(p. 2131).*
- class **RemoveInfoMarshaller**
  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2189).*
- class **RemoveSubscriptionInfoMarshaller**
  Marshaling code for Open Wire Format for ***RemoveSubscriptionInfoMarshaller*** *(p. 2202).*
- class **ReplayCommandMarshaller**
  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2218).*
- class **ResponseMarshaller**
  Marshaling code for Open Wire Format for ***ResponseMarshaller*** *(p. 2246).*
- class **SessionIdMarshaller**
  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(p. 2292).*
- class **SessionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(p. 2312).*
- class **ShutdownInfoMarshaller**
  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(p. 2360).*
- class **SubscriptionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2500).*
- class **TransactionIdMarshaller**
  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(p. 2581).*
- class **TransactionInfoMarshaller**
  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2598).*
- class **WireFormatInfoMarshaller**
  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(p. 2709).*
- class **XATransactionIdMarshaller**
  Marshaling code for Open Wire Format for ***XATransactionIdMarshaller*** *(p. 2740).*

5.23 activemq::wireformat::openwire::utils Namespace Reference

Data Structures

- class **BooleanStream**

  Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

- class **HexTable**

  *The **HexTable** (p. 1388) class maps hexidecimal strings to the value of an index into the table, i.e.*

- class **MessagePropertyInterceptor**

  Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

- class **OpenwireStringSupport**

5.24 activemq::wireformat::stomp Namespace Reference

Data Structures

- class **StompCommandConstants**
- class **StompFrame**

  A Stomp-level message frame that encloses all messages to and from the broker.

- class **StompHelper**

  Utility Methods used when marshaling to and from StompFrame's.

- class **StompWireFormat**
- class **StompWireFormatFactory**

  Factory used to create the Stomp Wire Format instance.

5.25 cms Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Data Structures

- class **BytesMessage**

  *A **BytesMessage** (p. 759) object is used to send a message containing a stream of unsigned bytes.*

- class **Closeable**

  Interface for a class that implements the close method.

- class **CMSException**

  CMS API Exception that is the base for all exceptions thrown from CMS classes.

- class **CMSProperties**

  Interface for a Java-like properties object.

- class **CMSSecurityException**

  This exception must be thrown when a provider rejects a user name/password submitted by a client.

- class **Connection**

  The client's connection to its provider.

- class **ConnectionFactory**

  *Defines the interface for a factory that creates connection objects, the **Connection** (p. 941) objects returned implement the CMS **Connection** (p. 941) interface and hide the CMS Provider specific implementation details behind that interface.*

- class **ConnectionMetaData**

  *A **ConnectionMetaData** (p. 1015) object provides information describing the **Connection** (p. 941) object.*

- class **DeliveryMode**

  This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

- class **Destination**

  *A **Destination** (p. 1190) object encapsulates a provider-specific address.*

- class **ExceptionListener**

  If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1275) that is registered with the `Connection` (p. 941).

- class **IllegalStateException**

  This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

- class **InvalidClientIdException**

  This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

- class **InvalidDestinationException**

  This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

- class **InvalidSelectorException**

  This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

- class **MapMessage**

  *A **MapMessage** (p. 1701) object is used to send a set of name-value pairs.*

- class **Message**

  Root of all messages.

- class **MessageConsumer**

  A client uses a `MessageConsumer` (p. 1795) to received messages from a destination.

- class **MessageEOFException**

  *This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2476) or **BytesMessage** (p. 759) is being read.*

- class **MessageFormatException**

  This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

- class **MessageListener**

  A `MessageListener` (p. 1860) object is used to receive asynchronously delivered messages.

- class **MessageNotReadableException**

  This exception must be thrown when a CMS client attempts to read a write-only message.

- class **MessageNotWriteableException**

  This exception must be thrown when a CMS client attempts to write to a read-only message.

- class **MessageProducer**

  *A client uses a `MessageProducer` (p. 1878) object to send messages to a **Destination** (p. 1190).*

- class **ObjectMessage**

  Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

- class **Queue**

  An interface encapsulating a provider-specific queue name.

- class **QueueBrowser**

  *This class implements in interface for browsing the messages in a **Queue** (p. 2157) without removing them.*

- class **Session**

  *A **Session** (p. 2270) object is a single-threaded context for producing and consuming messages.*

- class **Startable**

  Interface for a class that implements the start method.

- class **Stoppable**

  Interface for a class that implements the stop method.

- class **StreamMessage**

  *Interface for a **StreamMessage** (p. 2476).*

- class **TemporaryQueue**

  *Defines a Temporary **Queue** (p. 2157) based **Destination** (p. 1190).*

- class **TemporaryTopic**

  *Defines a Temporary **Topic** (p. 2571) based **Destination** (p. 1190).*

- class **TextMessage**

  Interface for a text message.

- class **Topic**

  An interface encapsulating a provider-specific topic name.

5.25.1 Detailed Description

5.26 decaf Namespace Reference

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

Namespaces

- namespace **internal**
- namespace **io**
- namespace **lang**
- namespace **net**
- namespace **nio**
- namespace **security**
- namespace **security_provider**
- namespace **util**

5.26.1 Detailed Description

Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements. See the NOTICE file distributed with this work for additional information regarding copyright ownership. The ASF licenses this file to You under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

`http://www.apache.org/licenses/LICENSE-2.0`

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

5.27 decaf::internal Namespace Reference

Namespaces

- namespace **io**
- namespace **net**
- namespace **nio**
- namespace **util**

Data Structures

- class **AprPool**

  *Wraps an APR pool object so that classes in **decaf** (p. 94) can create a static member for use in static methods where apr function calls that need a pool are made.*

- class **DecafRuntime**

  Handles APR initialization and termination.

5.28 decaf::internal::io Namespace Reference

Data Structures

- class **StandardErrorOutputStream**

  Wrapper Around the Standard error Output facility on the current platform.

- class **StandardInputStream**
- class **StandardOutputStream**

5.29 decaf::internal::net Namespace Reference

Data Structures

- class **URIEncoderDecoder**
- class **URIHelper**

  Helper class used by the URI classes in encoding and decoding of URI's.

- class **URIType**

  Basic type object that holds data that composes a given URI.

5.30 decaf::internal::nio Namespace Reference

Data Structures

- class **BufferFactory**

  Factory class used by static methods in the ***decaf::nio*** *(p. 106) package to create the various default version of the NIO interfaces.*

- class **ByteArrayBuffer**

  This class defines six categories of operations upon byte buffers:.

- class **ByteArrayPerspective**

  This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective.

- class **CharArrayBuffer**
- class **DoubleArrayBuffer**
- class **FloatArrayBuffer**
- class **IntArrayBuffer**
- class **LongArrayBuffer**
- class **ShortArrayBuffer**

5.31 decaf::internal::util Namespace Reference

Data Structures

- class **ByteArrayAdapter**

  This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- class **HexStringParser**

5.32 decaf::io Namespace Reference

Data Structures

- class **BlockingByteArrayInputStream**

  This is a blocking version of a byte buffer stream.

- class **BufferedInputStream**

  A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of ***io*** *(p. 100) operations on the input stream.*

- class **BufferedOutputStream**

  Wrapper around another output stream that buffers output before writing to the target output stream.

- class **ByteArrayInputStream**

  Simple implementation of ***InputStream*** *(p. 1406) that wraps around an STL Vector std::vector<unsigned char>.*

- class **ByteArrayOutputStream**
- class **Closeable**

  Interface for a class that implements the close method.

- class **DataInputStream**

  A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

- class **DataOutputStream**

  A data output stream lets an application write primitive Java data types to an output stream in a portable way.

- class **EOFException**
- class **FilterInputStream**

  A ***FilterInputStream*** *(p. 1315) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

- class **FilterOutputStream**

  This class is the superclass of all classes that filter output streams.

- class **InputStream**

  Base interface for an input stream.

- class **InterruptedIOException**
- class **IOException**
- class **OutputStream**

  Base interface for an output stream.

- class **Reader**
- class **UTFDataFormatException**

  Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

- class **Writer**

5.33 decaf::lang Namespace Reference

Namespaces

- namespace **exceptions**

Data Structures

- class **Appendable**
- class **Boolean**
- class **Byte**
- class **Character**
- class **CharSequence**

  *A **CharSequence** (p. 833) is a readable sequence of char values.*

- class **Comparable**

  This interface imposes a total ordering on the objects of each class that implements it.

- class **Double**
- class **Exception**
- class **Float**
- class **Integer**
- class **Iterable**

  *Implementing this interface allows an object to be cast to an **Iterable** (p. 1487) type for generic collections API calls.*

- class **Long**
- class **Math**

  *The class **Math** (p. 1718) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

- class **Number**

  *The abstract class **Number** (p. 1954) is the superclass of classes **Byte** (p. 664), **Double** (p. 1231), **Float** (p. 1328), **Integer** (p. 1428), **Long** (p. 1652), and **Short** (p. 2321).*

- class **AtomicRefCounter**
- struct **STATIC_CAST_TOKEN**
- struct **DYNAMIC_CAST_TOKEN**
- class **Pointer**

  *Decaf's implementation of a Smart **Pointer** (p. 2011) that is a template on a Type and is **Thread** (p. 2544) Safe if the default Reference Counter is used.*

- class **PointerComparator**

  *This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2011) instance.*

- class **Runnable**

  Interface for a runnable object - defines a task that can be run by a thread.

- class **Runtime**
- class **Short**
- class **System**
- class **Thread**

  *Basic thread class - mimics the Java **Thread** (p. 2544).*

- class **Throwable**

  This class represents an error that has occurred.

Functions

- template<typename T , typename R , typename U >
  bool **operator==** (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
  bool **operator==** (const U *left, const **Pointer**< T, R > &right)
- template<typename T , typename R , typename U >
  bool **operator!=** (const **Pointer**< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
  bool **operator!=** (const U *left, const **Pointer**< T, R > &right)

5.33.1 Function Documentation

5.33.1.1 template<typename T , typename R , typename U > bool decaf::lang::operator!= (const U * *left*, const Pointer< T, R > & *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.33.1.2 template<typename T , typename R , typename U > bool decaf::lang::operator!= (const Pointer< T, R > & *left*, const U * *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.33.1.3 template<typename T , typename R , typename U > bool decaf::lang::operator== (const U * *left*, const Pointer< T, R > & *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.33.1.4 template<typename T , typename R , typename U > bool decaf::lang::operator== (const Pointer< T, R > & *left*, const U * *right*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

5.34 decaf::lang::exceptions Namespace Reference

Data Structures

- class **ClassCastException**
- class **IllegalArgumentException**
- class **IllegalMonitorStateException**
- class **IllegalStateException**
- class **IndexOutOfBoundsException**
- class **InterruptedException**
- class **InvalidStateException**
- class **NoSuchElementException**
- class **NullPointerException**
- class **NumberFormatException**
- class **RuntimeException**
- class **UnsupportedOperationException**

5.35 decaf::net Namespace Reference

Data Structures

- class **BindException**
- class **BufferedSocket**

  *Buffered **Socket** (p. 2371) class that wraps a `Socket` (p. 2371) derived object and provides Buffered input and Output Streams to improce the efficiency of the reads and writes.*

- class **ConnectException**
- class **HttpRetryException**
- class **MalformedURLException**
- class **NoRouteToHostException**
- class **PortUnreachableException**
- class **ProtocolException**
- class **ServerSocket**

  A server socket class (for testing purposes).

- class **Socket**
- class **SocketError**

  Static utility class to simplify handling of error codes for socket operations.

- class **SocketException**

  Exception for errors when manipulating sockets.

- class **SocketFactory**

  ***Socket** (p. 2371) Factory implementation for use in Creating Sockets.*

- class **SocketInputStream**

  Input stream for performing reads on a socket.

- class **SocketOutputStream**

  Output stream for performing write operations on a socket.

- class **SocketTimeoutException**
- class **TcpSocket**

  Platform-independent implementation of the socket interface.

- class **UnknownHostException**
- class **UnknownServiceException**
- class **URI**

  *This class represents an instance of a **URI** (p. 2638) as defined by RFC 2396.*

- class **URISyntaxException**
- class **URL**

  *Class **URL** (p. 2677) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

- class **URLDecoder**
- class **URLEncoder**

5.36 decaf::nio Namespace Reference

Data Structures

- class **Buffer**

  A container for data of a specific primitive type.

- class **BufferOverflowException**
- class **BufferUnderflowException**
- class **ByteBuffer**

  This class defines six categories of operations upon byte buffers:.

- class **CharBuffer**

  This class defines four categories of operations upon character buffers:.

- class **DoubleBuffer**

  This class defines four categories of operations upon double buffers:.

- class **FloatBuffer**

  This class defines four categories of operations upon float buffers:.

- class **IntBuffer**

  This class defines four categories of operations upon int buffers:.

- class **InvalidMarkException**
- class **LongBuffer**

  This class defines four categories of operations upon long long buffers:.

- class **ReadOnlyBufferException**
- class **ShortBuffer**

  This class defines four categories of operations upon short buffers:.

5.37 decaf::security Namespace Reference

Namespaces

- namespace **auth**
- namespace **cert**

Data Structures

- class **GeneralSecurityException**
- class **InvalidKeyException**
- class **Key**

  *The **Key** (p. 1570) interface is the top-level interface for all keys.*

- class **KeyException**
- class **NoSuchAlgorithmException**
- class **NoSuchProviderException**
- class **Principal**

  Base interface for a principal, which can represent an individual or organization.

- class **PublicKey**

  A public key.

- class **SignatureException**

5.38 decaf::security::auth Namespace Reference

Namespaces

- namespace **x500**

5.39 decaf::security::auth::x500 Namespace Reference

Data Structures

- class **X500Principal**

5.40 decaf::security::cert Namespace Reference

Data Structures

- class **Certificate**

  Base interface for all identity certificates.

- class **CertificateEncodingException**
- class **CertificateException**
- class **CertificateExpiredException**
- class **CertificateNotYetValidException**
- class **CertificateParsingException**
- class **X509Certificate**

  Base interface for all identity certificates.

5.41 decaf::security_provider Namespace Reference

Namespaces

- namespace **unix**

Data Structures

- class **SecurityProvider**
- class **SecurityProviderMap**

  Lookup Map for Connector Factories.

- class **SecurityProviderRegistrar**

  Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

5.42 decaf::security_provider::unix Namespace Reference

Namespaces

- namespace **openssl**

5.43 decaf::security_provider::unix::openssl Namespace Reference

Data Structures

- class **OpenSSLX500Principal**

  The `OpenSSLX500Principal` (p. 1961) wraps around an OpenSSL `X509_NAME` structure.

- class **OpenSSLX509Certificate**

5.44 decaf::util Namespace Reference

Namespaces

- namespace **concurrent**
- namespace **logging**

Data Structures

- class **AbstractCollection**

  *This class provides a skeletal implementation of the **Collection** (p. 871) interface, to minimize the effort required to implement this interface.*

- class **AbstractList**

  *This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

- class **AbstractMap**

  *This class provides a skeletal implementation of the **Map** (p. 1689) interface, to minimize the effort required to implement this interface.*

- class **AbstractQueue**

  *This class provides skeletal implementations of some **Queue** (p. 2154) operations.*

- class **AbstractSequentialList**

  *This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

- class **AbstractSet**

  *This class provides a skeletal implementation of the **Set** (p. 2320) interface to minimize the effort required to implement this interface.*

- class **Collection**

  The root interface in the collection hierarchy.

- class **Comparator**

  A comparison function, which imposes a total ordering on some collection of objects.

- class **Date**

  Wrapper class around a time value in milliseconds.

- class **Iterator**

  Defines an object that can be used to iterate over the elements of a collection.

- class **List**

  An ordered collection (also known as a sequence).

- class **ListIterator**

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

- class **Map**

  Map *(p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **Properties**

  Java-like properties class for mapping string names to string values.

- class **Queue**

  A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

- class **Random**

  Random *(p. 2160) Value Generator which is used to generate a stream of pseudorandom numbers.*

- class **Set**

  A collection that contains no duplicate elements.

- class **StlList**

  List *(p. 1591) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **StlMap**

  Map *(p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **StlQueue**

  The ***Queue*** *(p. 2154) class accepts messages with an psuh(m) command where m is the message to be queued.*

- class **StlSet**

  Set *(p. 2320) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.*

- class **StringTokenizer**
- class **UUID**

  *A class that represents an immutable universally unique identifier (****UUID*** *(p. 2686)).*

5.45 decaf::util::concurrent Namespace Reference

Namespaces

- namespace **atomic**
- namespace **locks**

Data Structures

- class **BrokenBarrierException**
- class **Callable**
- class **CancellationException**
- class **ConcurrentMap**

  *Interface for a **Map** (p. 1689) type that provides additional **atomic** (p. 118) putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1689) interface.*

- class **ConcurrentStlMap**

  ***Map** (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **CountDownLatch**
- class **Delayed**

  A mix-in style interface for marking objects that should be acted upon after a given delay.

- class **ExecutionException**
- class **Executor**

  *An object that executes submitted **decaf.lang.Runnable** (p. 2256) tasks.*

- class **Future**

  *A **Future** (p. 1374) represents the result of an asynchronous computation.*

- class **Lock**

  A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

- class **Mutex**

  Creates a pthread_mutex_t object.

- class **PooledThread**
- class **PooledThreadListener**

  Abstract Listener Interface for users of `ThreadPool` (p. 2549).

- class **RejectedExecutionException**
- class **Synchronizable**

  The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

- class **TaskListener**
- class **ThreadFactory**

  *public interface **ThreadFactory** (p. 2547)*

- class **ThreadPool**

  Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

- class **TimeoutException**
- class **TimeUnit**

  *A **TimeUnit** (p. 2562) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

5.46 decaf::util::concurrent::atomic Namespace Reference

Data Structures

- class **AtomicBoolean**

  A boolean value that may be updated atomically.

- class **AtomicInteger**

  An int value that may be updated atomically.

- class **AtomicReference**

  An Pointer reference that may be updated atomically.

5.47 decaf::util::concurrent::locks Namespace Reference

Data Structures

- class **Condition**

  Condition *(p. 932) factors out the* ***Mutex*** *(p. 1921) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary* ***Lock*** *(p. 1618) implementations.*

- class **Lock**

  Lock *(p. 1618) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

- class **ReadWriteLock**

  A ***ReadWriteLock*** *(p. 2170) maintains a pair of associated* ***locks*** *(p. 119), one for read-only operations and one for writing.*

5.48 decaf::util::logging Namespace Reference

Data Structures

- class **Filter**

  *A **Filter** (p. 1314) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

- class **Formatter**

  *A **Formatter** (p. 1372) provides support for formatting LogRecords.*

- class **Handler**

  *A **Handler** (p. 1382) object takes log messages from a **Logger** (p. 1623) and exports them.*

- class **Logger**
- class **LoggerHierarchy**
- class **LogManager**

  *There is a single global **LogManager** (p. 1640) object that is used to maintain a set of shared state about Loggers and log services.*

- class **LogRecord**
- class **LogWriter**
- class **MarkBlockLogger**

  Defines a class that can be used to mark the entry and exit from scoped blocks.

- class **PropertiesChangeListener**

  *Defines the interface that classes can use to listen for change events on **Properties** (p. 2140).*

- class **SimpleFormatter**

  *Print a brief summary of the **LogRecord** (p. 1645) in a human readable format.*

- class **SimpleLogger**
- class **StreamHandler**

Enumerations

- enum **Level** {
  Off, **Null**, **Markblock**, **Debug**,
  Info, **Warn**, **Error**, **Fatal**,
  Throwing }

  Defines an enumeration for logging levels.

5.48.1 Enumeration Type Documentation

5.48.1.1 enum decaf::util::logging::Level

Defines an enumeration for **logging** (p. 120) levels.

Enumerator:

Off

Null

Markblock

Debug

Info

Warn

Error

Fatal

Throwing

5.49 std Namespace Reference

Data Structures

- struct **less< decaf::lang::Pointer< T > >**

  An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Chapter 6

Data Structure Documentation

6.1 decaf::util::AbstractCollection< E > Class Template Reference

This class provides a skeletal implementation of the **Collection** (p. 871) interface, to minimize the effort required to implement this interface.

`#include <src/main/decaf/util/AbstractCollection.h>`Inheritance diagram for decaf::util::AbstractCollection< E >:

Public Member Functions

- virtual ~**AbstractCollection** ()
- **AbstractCollection**< E > & **operator**= (const **AbstractCollection**< E > &collection)

  Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

- virtual bool **add** (const E &value DECAF_UNUSED) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

  Ensures that this collection contains the specified element (optional operation).

- virtual bool **addAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

  Adds all of the elements in the specified collection to this collection (optional operation).

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

  Removes all of the elements from this collection (optional operation).

- virtual void **copy** (const **Collection**< E > &collection)

  *Renders this **Collection** (p. 871) as a Copy of the given **Collection** (p. 871).*

- virtual bool **contains** (const E &value) const throw (lang::Exception)

  Returns true if this collection contains the specified element.

- virtual bool **containsAll** (const **Collection**< E > &collection) const throw (lang::Exception)

  Returns true if this collection contains all of the elements in the specified collection.

- virtual bool **equals** (const **Collection**< E > &collection) const

  *Answers true if this **Collection** (p. 871) and the one given are the same size and if each element contained in the **Collection** (p. 871) given is equal to an element contained in this collection.*

- virtual bool **isEmpty** () const

  Returns true if this collection contains no elements.

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes a single instance of the specified element from this collection, if it is present (optional operation).

- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes all of this collection's elements that are also contained in the specified collection (optional operation).

- virtual bool **retainAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Retains only the elements in this collection that are contained in the specified collection (optional operation).

- virtual std::vector< E > **toArray** () const

  *Answers an STL vector containing copies of all elements contained in this **Collection** (p. 871).*

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

Protected Attributes

- **util::concurrent::Mutex mutex**

6.1.1 Detailed Description

template<typename E> class decaf::util::AbstractCollection< E >

This class provides a skeletal implementation of the **Collection** (p. 871) interface, to minimize the effort required to implement this interface. To implement an unmodifiable collection, the programmer needs only to extend this class and provide implementations for the iterator and size methods. (The iterator returned by the iterator method must implement hasNext and next.)

To implement a modifiable collection, the programmer must additionally override this class's add method (which otherwise throws an UnsupportedOperationException), and the iterator returned by the iterator method must additionally implement its remove method.

The programmer should generally provide a void (no argument) and **Collection** (p. 871) constructor, as per the recommendation in the **Collection** (p. 871) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since:

1.0

6.1.2 Constructor & Destructor Documentation

6.1.2.1 template<typename E> virtual decaf::util::AbstractCollection< E >::~AbstractCollection () [inline, virtual]

6.1.3 Member Function Documentation

6.1.3.1 template<typename E> virtual bool decaf::util::AbstractCollection< E >::add (const E &value *DECAF_UNUSED*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]

Ensures that this collection contains the specified element (optional operation). Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. **Collection** (p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

This implementation always throws an UnsupportedOperationException.

Parameters:

value - The element that must be ensured to be in this collection.

Returns:

true if the collection was changed as a result of this call.

Exceptions:

UnsupportedOperationException if the add operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::addAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=().

6.1.3.2 template<typename E> virtual bool decaf::util::AbstractCollection< E >::addAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]

Adds all of the elements in the specified collection to this collection (optional operation). The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

This implementation iterates over the specified collection, and adds each object returned by the iterator to this collection, in turn.

Note that this implementation will throw an UnsupportedOperationException unless add is overridden (assuming the specified collection is non-empty).

Parameters:

collection - The **Collection** (p. 871) whose elements are to be added to this **Collection** (p. 871).

Returns:

true if the collection was changed as a result of this call.

Exceptions:

UnsupportedOperationException if the addAll operation is not supported by this collection

IllegalArgumentException if some property of the element prevents it from being added to this collection

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 137).

6.1.3.3 template<typename E> virtual void decaf::util::AbstractCollection< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all of the elements from this collection (optional operation). The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1490) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Implements **decaf::util::Collection< E >** (p. 874).

Reimplemented in **decaf::util::AbstractQueue< E >** (p. 138), **decaf::util::StlList< E >** (p. 2423), **decaf::util::StlSet< E >** (p. 2449), **decaf::util::StlList< CompositeTask * >** (p. 2423), **decaf::util::StlList< URI >** (p. 2423), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2423), **decaf::util::StlList< PrimitiveValueNode >** (p. 2423), **decaf::util::StlList< Pointer< Command > >** (p. 2423), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2423), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2449), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2449).

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=().

6.1.3.4 template<typename E> virtual bool decaf::util::AbstractCollection< E >::contains (const E & *value*) const throw (lang::Exception) [inline, virtual]

Returns true if this collection contains the specified element. This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 871).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Implements **decaf::util::Collection< E >** (p. 875).

Reimplemented in **decaf::util::StlList< E >** (p. 2423), **decaf::util::StlSet< E >** (p. 2450), **decaf::util::StlList< CompositeTask * >** (p. 2423), **decaf::util::StlList< URI >** (p. 2423), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2423), **decaf::util::StlList< PrimitiveValueNode >** (p. 2423), **decaf::util::StlList< Pointer< Command > >** (p. 2423),

decaf::util::StlList< Pointer< BackupTransport > > (p. 2423), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2450), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2450).

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::containsAll().

6.1.3.5 template<typename E> virtual bool decaf::util::AbstractCollection< E >::containsAll (const Collection< E > & *collection*) const throw (lang::Exception) [inline, virtual]

Returns true if this collection contains all of the elements in the specified collection. This implementation iterates over the specified collection, checking each element returned by the iterator in turn to see if it's contained in this collection. If all elements are so contained true is returned, otherwise false.

Parameters:

collection collection to be checked for containment in this collection

Returns:

true if this collection contains all of the elements in the specified collection.

Exceptions:

Exception if an error occurs,

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals().

6.1.3.6 template<typename E> virtual void decaf::util::AbstractCollection< E >::copy (const Collection< E > & *collection*) [inline, virtual]

Renders this **Collection** (p. 871) as a Copy of the given **Collection** (p. 871). This implementation iterates over the contents of the given collection adding each to this collection after first calling this Collection's clear method.

Parameters:

collection - the collection to mirror.

6.1.3.7 template<typename E> virtual bool decaf::util::AbstractCollection< E >::equals (const Collection< E > & *collection*) const [inline, virtual]

Answers true if this **Collection** (p. 871) and the one given are the same size and if each element contained in the **Collection** (p. 871) given is equal to an element contained in this collection.

Parameters:

collection - The **Collection** (p. 871) to be compared to this one.

Returns:

true if this **Collection** (p. 871) is equal to the one given.

6.1.3.8 template<typename E> virtual bool decaf::util::AbstractCollection< E >::isEmpty () const [inline, virtual]

Returns true if this collection contains no elements. This implementation returns **size()** (p. 878) == 0.

Returns:

true if the size method return 0.

Implements **decaf::util::Collection< E >** (p. 876).

Reimplemented in **decaf::util::StlList< E >** (p. 2425), **decaf::util::StlSet< E >** (p. 2450), **decaf::util::StlList< CompositeTask * >** (p. 2425), **decaf::util::StlList< URI >** (p. 2425), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2425), **decaf::util::StlList< PrimitiveValueNode >** (p. 2425), **decaf::util::StlList< Pointer< Command > >** (p. 2425), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2425), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2450), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2450).

6.1.3.9 template<typename E> virtual void decaf::util::AbstractCollection< E >::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.1.3.10 template<typename E> virtual void decaf::util::AbstractCollection< E >::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.1.3.11 template<typename E> virtual void decaf::util::AbstractCollection< E >::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.1.3.12 template<typename E> AbstractCollection<E>& decaf::util::AbstractCollection< E >::operator= (const AbstractCollection< E > & *collection*) [inline]

Assignment Operator, copy element from the source collection to this collection after clearing any element stored in this collection.

Parameters:

collection - the collection to copy

Returns:

a reference to this collection

6.1.3.13 template<typename E> virtual bool decaf::util::AbstractCollection< E >::remove (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes a single instance of the specified element from this collection, if it is present (optional operation). More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 871).

Implements **decaf::util::Collection< E >** (p. 876).

Reimplemented in **decaf::util::StlList< E >** (p. 2427), **decaf::util::StlSet< E >** (p. 2451), **decaf::util::StlList< CompositeTask * >** (p. 2427), **decaf::util::StlList< URI >** (p. 2427), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2427), **decaf::util::StlList< PrimitiveValueNode >** (p. 2427), **decaf::util::StlList< Pointer< Command > >** (p. 2427), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2427), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2451), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2451).

6.1.3.14 template<typename E> virtual bool decaf::util::AbstractCollection< E >::removeAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes all of this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements in common with the specified collection.

Parameters:

collection - collection containing elements to be removed from this collection

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

Reimplemented in **decaf::util::AbstractSet< E >** (p. 141), **decaf::util::AbstractSet< Pointer< Synchronization > >** (p. 141), and **decaf::util::AbstractSet< ActiveMQSession * >** (p. 141).

6.1.3.15 template<typename E> virtual bool decaf::util::AbstractCollection< E >::retainAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

This implementation iterates over this collection, checking each element returned by the iterator in turn to see if it's contained in the specified collection. If it's not so contained, it's removed from this collection with the iterator's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method and this collection contains one or more elements not present in the specified collection.

Parameters:

collection - collection containing elements to be retained in this collection

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection

IllegalArgumentException.

6.1.3.16 template<typename E> virtual std::vector<E> decaf::util::AbstractCollection< E >::toArray () const [inline, virtual]

Answers an STL vector containing copies of all elements contained in this **Collection** (p. 871). All the elements in the array will not be referenced by the collection. The elements in the returned array will be sorted to the same order as those returned by the iterator of this collection itself if the collection guarantees the order.

Returns:

an vector of copies of all the elements from this **Collection** (p. 871)

Implements **decaf::util::Collection< E >** (p. 878).

6.1.3.17 template<typename E> virtual void decaf::util::AbstractCollection< E >::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.1.3.18 template<typename E> virtual void decaf::util::AbstractCollection< E >::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.1.3.19 template<typename E> virtual void decaf::util::AbstractCollection< E >::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.1.4 Field Documentation

6.1.4.1 template<typename E> util::concurrent::Mutex decaf::util::AbstractCollection< E >::mutex [mutable, protected]

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractCollection.h**

6.2 decaf::util::AbstractList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).

`#include <src/main/decaf/util/AbstractList.h>`Inheritance diagram for decaf::util::AbstractList< E >:

Public Member Functions

- virtual ~**AbstractList** ()

6.2.1 Detailed Description

template<typename E> class decaf::util::AbstractList< E >

This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array). For sequential access data (such as a linked list), **AbstractSequentialList** (p. 140) should be used in preference to this class.

To implement an unmodifiable list, the programmer needs only to extend this class and provide implementations for the get(int) and **size()** (p. 878) methods.

To implement a modifiable list, the programmer must additionally override the set(int, E) method (which otherwise throws an UnsupportedOperationException). If the list is variable-size the programmer must additionally override the add(int, E) and remove(int) methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 871) interface specification.

Unlike the other abstract collection implementations, the programmer does not have to provide an iterator implementation; the iterator and list iterator are implemented by this class, on top of the "random access" methods: get(int), set(int, E), add(int, E) and remove(int).

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the collection being implemented admits a more efficient implementation.

Since:

1.0

6.2.2 Constructor & Destructor Documentation

6.2.2.1 template<typename E > virtual decaf::util::AbstractList< E >::~AbstractList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractList.h**

6.3 decaf::util::AbstractMap< K, V, COMPARATOR > Class Template Reference

This class provides a skeletal implementation of the **Map** (p. 1689) interface, to minimize the effort required to implement this interface.

`#include <src/main/decaf/util/AbstractMap.h>`Inheritance diagram for decaf::util::AbstractMap< K, V, COMPARATOR >:

Public Member Functions

- virtual ~**AbstractMap** ()

6.3.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::AbstractMap< K, V, COMPARATOR >

This class provides a skeletal implementation of the **Map** (p. 1689) interface, to minimize the effort required to implement this interface. To implement an unmodifiable map, the programmer needs only to extend this class and provide an implementation for the entrySet method, which returns a set-view of the map's mappings. Typically, the returned set will, in turn, be implemented atop **AbstractSet** (p. 141). This set should not support the add or remove methods, and its iterator should not support the remove method.

To implement a modifiable map, the programmer must additionally override this class's put method (which otherwise throws an UnsupportedOperationException), and the iterator returned by entrySet().iterator() must additionally implement its remove method.

The programmer should generally provide a void (no argument) and map constructor, as per the recommendation in the **Map** (p. 1689) interface specification.

The documentation for each non-abstract method in this class describes its implementation in detail. Each of these methods may be overridden if the map being implemented admits a more efficient implementation.

Since:

1.0

6.3.2 Constructor & Destructor Documentation

6.3.2.1 template<typename K , typename V , typename COMPARATOR > virtual decaf::util::AbstractMap< K, V, COMPARATOR >::~AbstractMap () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractMap.h**

6.4 decaf::util::AbstractQueue< E > Class Template Reference

This class provides skeletal implementations of some **Queue** (p. 2154) operations.

`#include <src/main/decaf/util/AbstractQueue.h>`Inheritance diagram for decaf::util::AbstractQueue< E >:

Public Member Functions

- **AbstractQueue** ()
- virtual ~**AbstractQueue** ()
- virtual bool **add** (const E &value) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

  Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

- virtual bool **addAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Adds all the elements of a collection to the queue.

- virtual E **remove** () throw (decaf::lang::exceptions::NoSuchElementException)

  Retrieves and removes the head of this queue.

- virtual const E & **element** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Retrieves, but does not remove, the head of this queue.

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

  Removes all elements of the queue.

6.4.1 Detailed Description

template<typename E> class decaf::util::AbstractQueue< E >

This class provides skeletal implementations of some **Queue** (p. 2154) operations. Methods add, remove, and element are based on offer, poll, and peek, respectively but throw exceptions instead of indicating failure via false or null returns.

A **Queue** (p. 2154) implementation that extends this class must minimally define a method **Queue** (p. 2154). offer(E) which does not permit insertion of null elements, along with methods **Queue** (p. 2154). **peek()** (p. 2156), **Queue.poll()** (p. 2156), **Collection.size()** (p. 878), and a **Collection.iterator()** (p. 1487) supporting **Iterator.remove()** (p. 1490). Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing **AbstractCollection** (p. 123).

Since:

1.0

6.4.2 Constructor & Destructor Documentation

6.4.2.1 template<typename E > decaf::util::AbstractQueue< E >::AbstractQueue () [inline]

6.4.2.2 template<typename E > virtual decaf::util::AbstractQueue< E >::~AbstractQueue () [inline, virtual]

6.4.3 Member Function Documentation

6.4.3.1 template<typename E > virtual bool decaf::util::AbstractQueue< E >::add (const E & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [inline, virtual]

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available. This implementation returns true if offer succeeds, else throws an IllegalStateException.

Parameters:

value - the element to offer to the **Queue** (p. 2154).

Returns:

true if the add succeeds.

Exceptions:

IllegalArgumentException if the element cannot be added.

Implements **decaf::util::Collection< E >** (p. 873).

References decaf::util::Queue< E >::offer().

6.4.3.2 template<typename E > virtual bool decaf::util::AbstractQueue< E >::addAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Adds all the elements of a collection to the queue. If the collection is the queue itself, then an IllegalArgumentException will be thrown out. If during the process, some runtime exception is thrown out, then part of the elements in the collection that have successfully added will remain in the queue.

The result of the method is undefined if the collection is modified during the process of the method.

Parameters:

collection - the collection to be added to the queue.

Returns:

true if the operation succeeds.

Exceptions:

IllegalArgumentException If the collection to be added to the queue is the queue itself.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 126).

6.4.3.3 template<typename E > virtual void decaf::util::AbstractQueue< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all elements of the queue. This implementation repeatedly invokes poll until it returns the empty marker.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 127).

References decaf::util::Queue< E >::getEmptyMarker(), and decaf::util::Queue< E >::poll().

6.4.3.4 template<typename E > virtual const E& decaf::util::AbstractQueue< E >::element () const throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]

Retrieves, but does not remove, the head of this queue. This method differs from peek only in that it throws an exception if this queue is empty.

This implementation returns the result of peek unless the queue is empty.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2155).

References decaf::util::Queue< E >::getEmptyMarker(), and decaf::util::Queue< E >::peek().

6.4.3.5 template<typename E > virtual E decaf::util::AbstractQueue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]

Retrieves and removes the head of this queue. This method differs from poll only in that it throws an exception if this queue is empty.

This implementation returns the result of poll unless the queue is empty.

Returns:

a copy of the element in the head of the queue.

Exceptions:

NoSuchElementException if the queue is empty.

Implements **decaf::util::Queue< E >** (p. 2156).

References decaf::util::Queue< E >::getEmptyMarker(), and decaf::util::Queue< E >::poll().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractQueue.h**

6.5 decaf::util::AbstractSequentialList< E > Class Template Reference

This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).

`#include <src/main/decaf/util/AbstractSequentialList.h>`Inheritance diagram for decaf::util::AbstractSequentialList< E >:

Public Member Functions

- virtual ~**AbstractSequentialList** ()

6.5.1 Detailed Description

template<typename E> class decaf::util::AbstractSequentialList< E >

This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list). For random access data (such as an array), **AbstractList** (p. 134) should be used in preference to this class.

This class is the opposite of the **AbstractList** (p. 134) class in the sense that it implements the "random access" methods (get(int index), set(int index, E element), add(int index, E element) and remove(int index)) on top of the list's list iterator, instead of the other way around.

To implement a list the programmer needs only to extend this class and provide implementations for the listIterator and size methods. For an unmodifiable list, the programmer need only implement the list iterator's hasNext, next, hasPrevious, previous and index methods.

For a modifiable list the programmer should additionally implement the list iterator's set method. For a variable-size list the programmer should additionally implement the list iterator's remove and add methods.

The programmer should generally provide a void (no argument) and collection constructor, as per the recommendation in the **Collection** (p. 871) interface specification.

Since:

1.0

6.5.2 Constructor & Destructor Documentation

6.5.2.1 template<typename E > virtual decaf::util::AbstractSequentialList< E >::~AbstractSequentialList () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSequentialList.h**

6.6 decaf::util::AbstractSet< E > Class Template Reference

This class provides a skeletal implementation of the **Set** (p. 2320) interface to minimize the effort required to implement this interface.

`#include <src/main/decaf/util/AbstractSet.h>`Inheritance diagram for decaf::util::AbstractSet< E >:

Public Member Functions

- virtual ~**AbstractSet** ()
- virtual bool **removeAll** (const **Collection**< E > &collection) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes from this set all of its elements that are contained in the specified collection (optional operation).

6.6.1 Detailed Description

template<typename E> class decaf::util::AbstractSet< E >

This class provides a skeletal implementation of the **Set** (p. 2320) interface to minimize the effort required to implement this interface. The process of implementing a set by extending this class is identical to that of implementing a **Collection** (p. 871) by extending **AbstractCollection** (p. 123), except that all of the methods and constructors in subclasses of this class must obey the additional constraints imposed by the **Set** (p. 2320) interface (for instance, the add method must not permit addition of multiple instances of an object to a set).

Since:

1.0

6.6.2 Constructor & Destructor Documentation

6.6.2.1 template<typename E> virtual decaf::util::AbstractSet< E >::~AbstractSet () [inline, virtual]

6.6.3 Member Function Documentation

6.6.3.1 template<typename E> virtual bool decaf::util::AbstractSet< E >::removeAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes from this set all of its elements that are contained in the specified collection (optional operation). If the specified collection is also a set, this operation effectively modifies this set so that its value is the asymmetric set difference of the two sets.

This implementation determines which is the smaller of this set and the specified collection, by invoking the size method on each. If this set has fewer elements, then the implementation iterates

over this set, checking each element returned by the iterator in turn to see if it is contained in the specified collection. If it is so contained, it is removed from this set with the iterator's remove method. If the specified collection has fewer elements, then the implementation iterates over the specified collection, removing from this set each element returned by the iterator, using this set's remove method.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by the iterator method does not implement the remove method.

Parameters:

collection - The **Collection** (p. 871) whose elements are to be retained

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 131).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**AbstractSet.h**

6.7 activemq::transport::AbstractTransportFactory Class Reference

Abstract implementation of the **TransportFactory** (p. 2614) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2614) instances.

`#include <src/main/activemq/transport/AbstractTransportFactory.h>`Inheritance diagram for activemq::transport::AbstractTransportFactory:

Public Member Functions

- virtual ~**AbstractTransportFactory** ()

Protected Member Functions

- virtual **Pointer**< **wireformat::WireFormat** > **createWireFormat** (const **decaf::util::Properties** &properties) throw (decaf::lang::exceptions::NoSuchElementException)

  *Creates the WireFormat that is configured for this **Transport** (p. 2608) and returns it.*

6.7.1 Detailed Description

Abstract implementation of the **TransportFactory** (p. 2614) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2614) instances.

Since:

3.0

6.7.2 Constructor & Destructor Documentation

6.7.2.1 virtual activemq::transport::AbstractTransportFactory::~AbstractTransportFactory () `[inline, virtual]`

6.7.3 Member Function Documentation

6.7.3.1 virtual Pointer<wireformat::WireFormat> activemq::transport::AbstractTransportFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::NoSuchElementException) `[protected, virtual]`

Creates the WireFormat that is configured for this **Transport** (p. 2608) and returns it. The default WireFormat is Openwire.

Parameters:

properties The properties that were configured on the URI.

Returns:

a pointer to a WireFormat instance that the caller then owns.

Exceptions:

NoSuchElementException if the configured WireFormat is not found.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**AbstractTransportFactory.h**

6.8 activemq::core::ActiveMQAckHandler Class Reference

Interface class that is used to give CMS Messages an interface to Ack themselves with.

`#include <src/main/activemq/core/ActiveMQAckHandler.h>`Inheritance diagram for activemq::core::ActiveMQAckHandler:

Public Member Functions

- virtual ~**ActiveMQAckHandler** ()
- virtual void **acknowledgeMessage** (const **commands::Message** *message)=0 throw (cms::CMSException)

  Method called to acknowledge the message passed.

6.8.1 Detailed Description

Interface class that is used to give CMS Messages an interface to Ack themselves with.

Since:

2.0

6.8.2 Constructor & Destructor Documentation

6.8.2.1 virtual activemq::core::ActiveMQAckHandler::~ActiveMQAckHandler () [inline, virtual]

6.8.3 Member Function Documentation

6.8.3.1 virtual void activemq::core::ActiveMQAckHandler::acknowledgeMessage (const commands::Message * *message*) throw (cms::CMSException) [pure virtual]

Method called to acknowledge the message passed.

Parameters:

message Message to Acknowledge

Exceptions:

CMSException

Implemented in **activemq::core::ActiveMQConsumer** (p. 221).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQAckHandler.h**

6.9 activemq::commands::ActiveMQBlobMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQBlobMessage.h>`Inheritance diagram for activemq::commands::ActiveMQBlobMessage:

Public Member Functions

- **ActiveMQBlobMessage** ()
- virtual ~**ActiveMQBlobMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQBlobMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::Message** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

- std::string **getRemoteBlobUrl** () const

  Get the Remote URL of the Blob.

- void **setRemoteBlobUrl** (const std::string &remoteURL)

  Set the Remote URL of the Blob.

- std::string **getMimeType** () const

  Get the Mime Type of the Blob.

- void **setMimeType** (const std::string &mimeType)

  Set the Mime Type of the Blob.

- std::string **getName** () const

  Gets the Name of the Blob.

- void **setName** (const std::string &name)

  Sets the Name of the Blob.

- bool **isDeletedByBroker** () const

  Gets if this Blob is deleted by the Broker.

- void **setDeletedByBroker** (bool value)

  Sets the Deleted By Broker flag.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQBLOBMESSAGE** = 29
- static const std::string **BINARY_MIME_TYPE**

6.9.1 Constructor & Destructor Documentation

6.9.1.1 activemq::commands::ActiveMQBlobMessage::ActiveMQBlobMessage ()

6.9.1.2 virtual activemq::commands::ActiveMQBlobMessage::~ActiveMQBlobMessage () [inline, virtual]

6.9.2 Member Function Documentation

6.9.2.1 virtual cms::Message* activemq::commands::ActiveMQBlobMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.9.2.2 virtual ActiveMQBlobMessage* activemq::commands::ActiveMQBlobMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.9.2.3 virtual void activemq::commands::ActiveMQBlobMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.9.2.4 virtual bool activemq::commands::ActiveMQBlobMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.9.2.5 virtual unsigned char activemq::commands::ActiveMQBlobMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.9.2.6 std::string activemq::commands::ActiveMQBlobMessage::getMimeType () const [inline]

Get the Mime Type of the Blob.

Returns:

string holding the MIME Type.

6.9.2.7 std::string activemq::commands::ActiveMQBlobMessage::getName () const [inline]

Gets the Name of the Blob.

Returns:

string name of the Blob.

6.9.2.8 std::string activemq::commands::ActiveMQBlobMessage::getRemoteBlobUrl () const [inline]

Get the Remote URL of the Blob.

Returns:

string from of the Remote Blob URL.

6.9.2.9 bool activemq::commands::ActiveMQBlobMessage::isDeletedByBroker () const [inline]

Gets if this Blob is deleted by the Broker.

Returns:

true if the Blob is deleted by the Broker.

6.9.2.10 void activemq::commands::ActiveMQBlobMessage::setDeletedByBroker (bool *value*) [inline]

Sets the Deleted By Broker flag.

Parameters:

value - set the Delete by broker flag to value.

6.9.2.11 void activemq::commands::ActiveMQBlobMessage::setMimeType (const std::string & *mimeType*) [inline]

Set the Mime Type of the Blob.

Parameters:

mimeType - String holding the MIME Type.

6.9.2.12 void activemq::commands::ActiveMQBlobMessage::setName (const std::string & *name*) [inline]

Sets the Name of the Blob.

Parameters:

name - Name of the Blob.

6.9.2.13 void activemq::commands::ActiveMQBlobMessage::setRemoteBlobUrl (const std::string & *remoteURL*) [inline]

Set the Remote URL of the Blob.

Parameters:

remoteURL - String form of the Remote URL.

6.9.2.14 virtual std::string activemq::commands::ActiveMQBlobMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.9.3 Field Documentation

6.9.3.1 const std::string activemq::commands::ActiveMQBlobMessage::BINARY_MIME_TYPE [static]

6.9.3.2 const unsigned char activemq::commands::ActiveMQBlobMessage::ID_ACTIVEMQBLOBMESSAGE = 29 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBlobMessage.h**

6.10 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 151).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual ~**ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.10.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 151). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.10.2 Constructor & Destructor Documentation

6.10.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::ActiveMQ () [inline]

6.10.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::~ActiveM () [inline, virtual]

6.10.3 Member Function Documentation

6.10.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::createObjec () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.10.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.10.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.10.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.10.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.10.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.10.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBlobMessageMarshaller.h**

6.11 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 155).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h>`Inheri diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual ~**ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.11.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 155). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.11.2 Constructor & Destructor Documentation

6.11.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::ActiveMQ () [inline]

6.11.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::~ActiveM () [inline, virtual]

6.11.3 Member Function Documentation

6.11.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::createObjec () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.11.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.11.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.11.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.11.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.11.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.11.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQBlobMessageMarshaller.h**

6.12 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 159).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBlobMessageMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller:

Public Member Functions

- **ActiveMQBlobMessageMarshaller** ()
- virtual ~**ActiveMQBlobMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.12.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 159). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.12.2 Constructor & Destructor Documentation

6.12.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::ActiveMQ () [inline]

6.12.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::~ActiveM () [inline, virtual]

6.12.3 Member Function Documentation

6.12.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::createObjec () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.12.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.12.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.12.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.12.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.12.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.12.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBlobMessageMarshaller.h**

6.13 activemq::commands::ActiveMQBytesMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQBytesMessage.h>`Inheritance diagram for activemq::commands::ActiveMQBytesMessage:

Public Member Functions

- **ActiveMQBytesMessage** ()
- virtual ~**ActiveMQBytesMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQBytesMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::BytesMessage** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual void **clearBody** () throw (cms::CMSException)

  Clears out the body of the message.

- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes) throw (cms::CMSException)

  sets the bytes given to the message body.

- virtual const unsigned char * **getBodyBytes** () const throw (cms::CMSException)

  Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

- virtual std::size_t **getBodyLength** () const throw (cms::CMSException)

  Returns the number of bytes contained in the body of this message.

- virtual void **reset** () throw (cms::CMSException)

  Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

- virtual bool **readBoolean** () const throw (cms::CMSException)

Reads a Boolean from the Bytes message stream.

- virtual void **writeBoolean** (bool value) throw (cms::CMSException)

  Writes a boolean to the bytes message stream as a 1-byte value.

- virtual unsigned char **readByte** () const throw (cms::CMSException)

  Reads a Byte from the Bytes message stream.

- virtual void **writeByte** (unsigned char value) throw (cms::CMSException)

  Writes a byte to the bytes message stream as a 1-byte value.

- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::CMSException)

  Reads a byte array from the bytes message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::CMSException)

  Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *&buffer, std::size_t length) const throw (cms::CMSException)

  Reads a portion of the bytes message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::CMSException)

  Writes a portion of a byte array to the bytes message stream.

- virtual char **readChar** () const throw (cms::CMSException)

  Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value) throw (cms::CMSException)

  Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const throw (cms::CMSException)

  Reads a 32 bit float from the Bytes message stream.

- virtual void **writeFloat** (float value) throw (cms::CMSException)

  Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const throw (cms::CMSException)

  Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value) throw (cms::CMSException)

  Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const throw (cms::CMSException)

  Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value) throw (cms::CMSException)

Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::CMSException)

  Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::CMSException)

  Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::CMSException)

  Reads a 32 bit signed intger from the Bytes message stream.

- virtual void **writeInt** (int value) throw (cms::CMSException)

  Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::CMSException)

  Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value) throw (cms::CMSException)

  Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::CMSException)

  Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value) throw (cms::CMSException)

  Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const throw (cms::CMSException)

  Reads an UTF String from the BytesMessage stream.

- virtual void **writeUTF** (const std::string &value) throw (cms::CMSException)

  Writes an UTF String to the BytesMessage stream.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQBYTESMESSAGE** = 24

Protected Member Functions

- void **checkWriteOnlyBody** () const throw (cms::CMSException)

  Throws an exception if in write-only mode.

6.13.1 Constructor & Destructor Documentation

6.13.1.1 activemq::commands::ActiveMQBytesMessage::ActiveMQBytesMessage ()

6.13.1.2 virtual activemq::commands::ActiveMQBytesMessage::~ActiveMQBytesMessage () [inline, virtual]

6.13.2 Member Function Documentation

6.13.2.1 void activemq::commands::ActiveMQBytesMessage::checkWriteOnlyBody () const throw (cms::CMSException) [protected]

Throws an exception if in write-only mode.

Exceptions:

CMSException.

6.13.2.2 virtual void activemq::commands::ActiveMQBytesMessage::clearBody () throw (cms::CMSException) [inline, virtual]

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297).

6.13.2.3 virtual cms::BytesMessage* activemq::commands::ActiveMQBytesMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::BytesMessage** (p. 762).

6.13.2.4 virtual ActiveMQBytesMessage* activemq::commands::ActiveMQBytesMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.13.2.5 virtual void activemq::commands::ActiveMQBytesMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.13.2.6 virtual bool activemq::commands::ActiveMQBytesMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Parameters:

value The **Command** (p. 879) to compare to this one.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.13.2.7 virtual const unsigned char* activemq::commands::ActiveMQBytesMessage::getBodyBytes () const throw (cms::CMSException) [virtual]

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer. Call `getBodyLength` to determine the number of bytes to expect.

Returns:

const pointer to a byte buffer

Implements **cms::BytesMessage** (p. 762).

6.13.2.8 virtual std::size_t activemq::commands::ActiveMQBytesMessage::getBodyLength () const throw (cms::CMSException) [virtual]

Returns the number of bytes contained in the body of this message.

Returns:

number of bytes.

Implements **cms::BytesMessage** (p. 762).

6.13.2.9 virtual unsigned char activemq::commands::ActiveMQBytesMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.13.2.10 virtual bool activemq::commands::ActiveMQBytesMessage::readBoolean () const throw (cms::CMSException) [virtual]

Reads a Boolean from the Bytes message stream.

Returns:

boolean value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 763).

6.13.2.11 virtual unsigned char activemq::commands::ActiveMQBytesMessage::readByte () const throw (cms::CMSException) [virtual]

Reads a Byte from the Bytes message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 763).

6.13.2.12 virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes (unsigned char *& *buffer*, std::size_t *length*) const throw (cms::CMSException) [virtual]

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than

the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an IndexOutOfBoundsException is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer - the buffer into which the data is read

length - the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 763).

6.13.2.13 virtual std::size_t activemq::commands::ActiveMQBytesMessage::readBytes (std::vector< unsigned char > & *value*) const throw (cms::CMSException) [virtual]

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value - buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException if an error occurs.

Implements **cms::BytesMessage** (p. 764).

6.13.2.14 virtual char activemq::commands::ActiveMQBytesMessage::readChar () const throw (cms::CMSException) [virtual]

Reads a Char from the Bytes message stream.

Returns:

char value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 764).

6.13.2.15 virtual double activemq::commands::ActiveMQBytesMessage::readDouble () const throw (cms::CMSException) [virtual]

Reads a 64 bit double from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 765).

6.13.2.16 virtual float activemq::commands::ActiveMQBytesMessage::readFloat () const throw (cms::CMSException) [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 765).

6.13.2.17 virtual int activemq::commands::ActiveMQBytesMessage::readInt () const throw (cms::CMSException) [virtual]

Reads a 32 bit signed intger from the Bytes message stream.

Returns:

int value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 765).

6.13.2.18 virtual long long activemq::commands::ActiveMQBytesMessage::readLong () const throw (cms::CMSException) [virtual]

Reads a 64 bit long from the Bytes message stream.

Returns:

long long value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 766).

6.13.2.19 virtual short activemq::commands::ActiveMQBytesMessage::readShort () const throw (cms::CMSException) [virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns:

short value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 766).

6.13.2.20 virtual std::string activemq::commands::ActiveMQBytesMessage::readString () const throw (cms::CMSException) [virtual]

Reads an ASCII String from the Bytes message stream.

Returns:

String from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 766).

6.13.2.21 virtual unsigned short activemq::commands::ActiveMQBytesMessage::readUnsignedShort () const throw (cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 767).

6.13.2.22 virtual std::string activemq::commands::ActiveMQBytesMessage::readUTF () const throw (cms::CMSException) [virtual]

Reads an UTF String from the BytesMessage stream.

Returns:

String from stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 767).

6.13.2.23 virtual void activemq::commands::ActiveMQBytesMessage::reset () throw (cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 767).

6.13.2.24 virtual void activemq::commands::ActiveMQBytesMessage::setBodyBytes (const unsigned char * *buffer*, std::size_t *numBytes*) throw (cms::CMSException) [virtual]

sets the bytes given to the message body.

Parameters:

buffer Byte Buffer to copy.

numBytes Number of bytes in Buffer to copy.

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 768).

6.13.2.25 virtual std::string activemq::commands::ActiveMQBytesMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.13.2.26 virtual void activemq::commands::ActiveMQBytesMessage::writeBoolean (bool *value*) throw (cms::CMSException) [virtual]

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value - boolean to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 768).

6.13.2.27 virtual void activemq::commands::ActiveMQBytesMessage::writeByte (unsigned char *value*) throw (cms::CMSException) [virtual]

Writes a byte to the bytes message stream as a 1-byte value.

Parameters:

value - byte to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 768).

6.13.2.28 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const unsigned char * *value*, std::size_t *offset*, std::size_t *length*) throw (cms::CMSException) [virtual]

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

Parameters:

value - bytes to write to the stream

offset - the initial offset within the byte array

length - the number of bytes to use

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 769).

6.13.2.29 virtual void activemq::commands::ActiveMQBytesMessage::writeBytes (const std::vector< unsigned char > & *value*) throw (cms::CMSException) [virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters:

value - bytes to write to the stream.

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 769).

6.13.2.30 virtual void activemq::commands::ActiveMQBytesMessage::writeChar (char *value*) throw (cms::CMSException) [virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters:

value - char to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 769).

6.13.2.31 virtual void activemq::commands::ActiveMQBytesMessage::writeDouble (double *value*) throw (cms::CMSException) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters:

value - double to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 770).

6.13.2.32 virtual void activemq::commands::ActiveMQBytesMessage::writeFloat (float *value*) throw (cms::CMSException) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters:

value - float to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 770).

6.13.2.33 virtual void activemq::commands::ActiveMQBytesMessage::writeInt (int *value*) throw (cms::CMSException) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters:

value - signed int to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 770).

6.13.2.34 virtual void activemq::commands::ActiveMQBytesMessage::writeLong (long long *value*) throw (cms::CMSException) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters:

value - signed long long to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 771).

6.13.2.35 virtual void activemq::commands::ActiveMQBytesMessage::writeShort (short *value*) throw (cms::CMSException) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters:

value - signed short to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 771).

6.13.2.36 virtual void activemq::commands::ActiveMQBytesMessage::writeString (const std::string & *value*) throw (cms::CMSException) [virtual]

Writes an ASCII String to the Bytes message stream.

Parameters:

value - String to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 771).

6.13.2.37 virtual void activemq::commands::ActiveMQBytesMessage::writeUnsignedShort (unsigned short *value*) throw (cms::CMSException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value - unsigned short to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 772).

6.13.2.38 virtual void activemq::commands::ActiveMQBytesMessage::writeUTF (const std::string & *value*) throw (cms::CMSException) [virtual]

Writes an UTF String to the BytesMessage stream.

Parameters:

value - String to write to the stream

Exceptions:

CMSException

Implements **cms::BytesMessage** (p. 772).

6.13.3 Field Documentation

6.13.3.1 const unsigned char activemq::commands::ActiveMQBytesMessage::ID_ACTIVEMQBYTESMESSAGE = 24 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQBytesMessage.h**

6.14 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessa Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 178).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBytesMessageMarshaller.h>`Inhe
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.14.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 178). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.14.2 Constructor & Destructor Documentation

6.14.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::ActiveMQ() [inline]

6.14.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::~ActiveM() [inline, virtual]

6.14.3 Member Function Documentation

6.14.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::createObje() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.14.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::getDataStr() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.14.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseMarsh(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.14.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.14.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.14.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.14.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller::tightUnmar (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQBytesMessageMarshaller.h**

6.15 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessa Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBytesMessageMarshaller.h>`Inhe
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.15.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.15.2 Constructor & Destructor Documentation

6.15.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::ActiveMQ () [inline]

6.15.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::~ActiveM () [inline, virtual]

6.15.3 Member Function Documentation

6.15.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::createObjec () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.15.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::getDataStr () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.15.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.15.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.15.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.15.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.15.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller::tightUnmar (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQBytesMessageMarshaller.h**

6.16 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessa Class Reference

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 186).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h>`Inhe
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller:

Public Member Functions

- **ActiveMQBytesMessageMarshaller** ()
- virtual ~**ActiveMQBytesMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.16.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 186). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.16.2 Constructor & Destructor Documentation

6.16.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::ActiveMQ() [inline]

6.16.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::~ActiveM() [inline, virtual]

6.16.3 Member Function Documentation

6.16.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::createObje() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.16.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::getDataStr() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.16.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseMarsh(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.16.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.16.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.16.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.16.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller::tightUnmar (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQBytesMessageMarshaller.h**

6.17 activemq::core::ActiveMQConnection Class Reference

Concrete connection used for all connectors to the ActiveMQ broker.

`#include <src/main/activemq/core/ActiveMQConnection.h>`Inheritance diagram for activemq::core::ActiveMQConnection:

Public Member Functions

- **ActiveMQConnection** (const **Pointer< transport::Transport >** &transport, const **Pointer< decaf::util::Properties >** &properties)

  Constructor.

- virtual ~**ActiveMQConnection** ()
- virtual void **removeSession** (**ActiveMQSession** *session) throw (cms::CMSException)

  Removes the session resources for the given session instance.

- virtual void **addProducer** (**ActiveMQProducer** *producer) throw (cms::CMSException)

  Adds an active Producer to the Set of known producers.

- virtual void **removeProducer** (const **Pointer< commands::ProducerId >** &producerId) throw (cms::CMSException)

  Removes an active Producer to the Set of known producers.

- virtual void **addDispatcher** (const **Pointer< commands::ConsumerId >** &consumer, **Dispatcher** *dispatcher) throw (cms::CMSException)

  Adds a dispatcher for a consumer.

- virtual void **removeDispatcher** (const **Pointer< commands::ConsumerId >** &consumer) throw (cms::CMSException)

  Removes the dispatcher for a consumer.

- virtual void **sendPullRequest** (const **commands::ConsumerInfo** *consumer, long long timeout) throw (exceptions::ActiveMQException)

  If supported sends a message pull request to the service provider asking for the delivery of a new message.

- bool **isClosed** () const

  Checks if this connection has been closed.

- bool **isStarted** () const

  Check if this connection has been started.

- virtual void **destroyDestination** (const **commands::ActiveMQDestination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)

Requests that the Broker removes the given Destination.

- virtual void **destroyDestination** (const **cms::Destination** *destination) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException)

  Requests that the Broker removes the given Destination.

- virtual const **cms::ConnectionMetaData** * **getMetaData** () const throw (cms::CMSException)

  Gets the metadata for this connection.

- virtual **cms::Session** * **createSession** () throw (cms::CMSException)

  Creates a new Session to work for this Connection.

- virtual **cms::Session** * **createSession** (**cms::Session::AcknowledgeMode** ackMode) throw (cms::CMSException)

  Creates a new Session to work for this Connection using the specified acknowledgment mode.

- virtual std::string **getClientID** () const

  Get the Client Id for this session.

- virtual void **close** () throw (cms::CMSException)

  Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

- virtual void **start** () throw (cms::CMSException)

  Starts or (restarts) a connections delivery of incoming messages.

- virtual void **stop** () throw (cms::CMSException)

  Stop the flow of incoming messages.

- virtual **cms::ExceptionListener** * **getExceptionListener** () const

  Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (**cms::ExceptionListener** *listener)

  Sets the registered Exception Listener for this connection.

- virtual void **onCommand** (const **Pointer**< **commands::Command** > &command)

  *Event handler for the receipt of a non-response command from the **transport** (p. 67).*

- virtual void **onException** (const **decaf::lang::Exception** &ex)

  *Event handler for an exception from a command **transport** (p. 67).*

- virtual void **transportInterrupted** ()

  *The **transport** (p. 67) has suffered an interruption from which it hopes to recover.*

- const **commands::ConnectionInfo** & **getConnectionInfo** () const throw (exceptions::ActiveMQException)

  Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

- const **commands::ConnectionId** & **getConnectionId** () const throw (exceptions::ActiveMQException)

  Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

- void **oneway** (**Pointer< commands::Command >** command) throw (activemq::exceptions::ActiveMQException)

  Sends a oneway message.

- void **syncRequest** (**Pointer< commands::Command >** command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)

  Sends a synchronous request and returns the response from the broker.

- void **disposeOf** (const **Pointer< commands::DataStructure >** &objectId) throw (activemq::exceptions::ActiveMQException)

  Sends a message to the broker to dispose of the given resource using an async oneway call.

- void **disposeOf** (const **Pointer< commands::DataStructure >** &objectId, unsigned int timeout) throw (activemq::exceptions::ActiveMQException)

  Sends a message to the broker to dispose of the given resource using a timed request.

- virtual void **fire** (const **exceptions::ActiveMQException** &ex)

  Notify the exception listener.

6.17.1 Detailed Description

Concrete connection used for all connectors to the ActiveMQ broker.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 activemq::core::ActiveMQConnection::ActiveMQConnection (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*)

Constructor.

Parameters:

transport *(*p. *67)* The Transport requested for this connection to the Broker.

properties The Properties that were defined for this connection

6.17.2.2 virtual activemq::core::ActiveMQConnection::~ActiveMQConnection () [virtual]

6.17.3 Member Function Documentation

6.17.3.1 virtual void activemq::core::ActiveMQConnection::addDispatcher (const Pointer< commands::ConsumerId > & *consumer*, Dispatcher * *dispatcher*) throw (cms::CMSException) [virtual]

Adds a dispatcher for a consumer.

Parameters:

consumer - The consumer for which to register a dispatcher.

dispatcher - The dispatcher to handle incoming messages for the consumer.

6.17.3.2 virtual void activemq::core::ActiveMQConnection::addProducer (ActiveMQProducer * *producer*) throw (cms::CMSException) [virtual]

Adds an active Producer to the Set of known producers.

Parameters:

producer - The Producer to add from the the known set.

6.17.3.3 virtual void activemq::core::ActiveMQConnection::close () throw (cms::CMSException) [virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions:

CMSException

Implements **cms::Connection** (p. 942).

6.17.3.4 virtual cms::Session* activemq::core::ActiveMQConnection::createSession (cms::Session::AcknowledgeMode *ackMode*) throw (cms::CMSException) [virtual]

Creates a new Session to work for this Connection using the specified acknowledgment mode.

Parameters:

ackMode the Acknowledgment Mode to use.

Exceptions:

CMSException

Implements **cms::Connection** (p. 942).

6.17.3.5 virtual cms::Session* activemq::core::ActiveMQConnection::createSession () throw (cms::CMSException) [virtual]

Creates a new Session to work for this Connection.

Exceptions:

CMSException

Implements **cms::Connection** (p. 943).

6.17.3.6 virtual void activemq::core::ActiveMQConnection::destroyDestination (const cms::Destination * *destination*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters:

destination The CMS Destination the Broker will be requested to remove.

Exceptions:

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.17.3.7 virtual void activemq::core::ActiveMQConnection::destroyDestination (const commands::ActiveMQDestination * *destination*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalStateException, decaf::lang::exceptions::UnsupportedOperationException, activemq::exceptions::ActiveMQException) [virtual]

Requests that the Broker removes the given Destination. Calling this method implies that the client is finished with the Destination and that no other messages will be sent or received for the given Destination. The Broker frees all resources it has associated with this Destination.

Parameters:

destination The Destination the Broker will be requested to remove.

Exceptions:

NullPointerException If the passed Destination is Null

IllegalStateException If the connection is closed.

UnsupportedOperationException If the wire format in use does not support this operation.

ActiveMQException If any other error occurs during the attempt to destroy the destination.

6.17.3.8 void activemq::core::ActiveMQConnection::disposeOf (const Pointer< commands::DataStructure > & *objectId*, unsigned int *timeout*) throw (activemq::exceptions::ActiveMQException)

Sends a message to the broker to dispose of the given resource using a timed request.

Parameters:

objectId The ID of the resource to be released.

timeout The time to wait for a response that the object is disposed.

Exceptions:

ConnectorException if any problems occur from sending the message.

6.17.3.9 void activemq::core::ActiveMQConnection::disposeOf (const Pointer< commands::DataStructure > & *objectId*) throw (activemq::exceptions::ActiveMQException)

Sends a message to the broker to dispose of the given resource using an async oneway call.

Parameters:

objectId The ID of the resource to be released.

Exceptions:

ConnectorException if any problems occur from sending the message.

6.17.3.10 virtual void activemq::core::ActiveMQConnection::fire (const exceptions::ActiveMQException & *ex*) [virtual]

Notify the exception listener.

Parameters:

ex the exception to fire

6.17.3.11 virtual std::string activemq::core::ActiveMQConnection::getClientID () const [virtual]

Get the Client Id for this session.

Returns:

string version of Client Id

Implements **cms::Connection** (p. 943).

6.17.3.12 const commands::ConnectionId& activemq::core::ActiveMQConnection::getConnectionId () const throw (exceptions::ActiveMQException)

Gets the ConnectionId for this Object, if the Connection is not open than this method throws an exception.

6.17.3.13 const commands::ConnectionInfo& activemq::core::ActiveMQConnection::getConnectionInfo () const throw (exceptions::ActiveMQException)

Gets the ConnectionInfo for this Object, if the Connection is not open than this method throws an exception.

6.17.3.14 virtual cms::ExceptionListener* activemq::core::ActiveMQConnection::getExceptionListener () const [inline, virtual]

Gets the registered Exception Listener for this connection.

Returns:

pointer to an exception listener or NULL

Implements **cms::Connection** (p. 943).

6.17.3.15 virtual const cms::ConnectionMetaData* activemq::core::ActiveMQConnection::getMetaData () const throw (cms::CMSException) [inline, virtual]

Gets the metadata for this connection.

Returns:

the connection MetaData pointer (caller does not own it).

Exceptions:

CMSException if the provider fails to get the connection metadata for this connection.

See also:

ConnectionMetaData

Since:

2.0

Implements **cms::Connection** (p. 943).

6.17.3.16 bool activemq::core::ActiveMQConnection::isClosed () const [inline]

Checks if this connection has been closed.

Returns:

true if the connection is closed

6.17.3.17 bool activemq::core::ActiveMQConnection::isStarted () const [inline]

Check if this connection has been started.

Returns:

true if the start method has been called.

6.17.3.18 virtual void activemq::core::ActiveMQConnection::onCommand (const Pointer< commands::Command > & *command*) [virtual]

Event handler for the receipt of a non-response command from the **transport** (p. 67).

Parameters:

command the received command object.

Implements **activemq::transport::TransportListener** (p. 2624).

6.17.3.19 void activemq::core::ActiveMQConnection::oneway (Pointer< commands::Command > *command*) throw (activemq::exceptions::ActiveMQException)

Sends a oneway message.

Parameters:

command The message to send.

Exceptions:

ConnectorException if not currently connected, or if the operation fails for any reason.

6.17.3.20 virtual void activemq::core::ActiveMQConnection::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

ex The exception.

Implements **activemq::transport::TransportListener** (p. 2625).

6.17.3.21 virtual void activemq::core::ActiveMQConnection::removeDispatcher (const Pointer< commands::ConsumerId > & *consumer*) throw (cms::CMSException) [virtual]

Removes the dispatcher for a consumer.

Parameters:

consumer - The consumer for which to remove the dispatcher.

6.17.3.22 virtual void activemq::core::ActiveMQConnection::removeProducer (const Pointer< commands::ProducerId > & *producerId*) throw (cms::CMSException) [virtual]

Removes an active Producer to the Set of known producers.

Parameters:

producerId - The ProducerId to remove from the the known set.

6.17.3.23 virtual void activemq::core::ActiveMQConnection::removeSession (ActiveMQSession * *session*) throw (cms::CMSException) [virtual]

Removes the session resources for the given session instance.

Parameters:

session The session to be unregistered from this connection.

6.17.3.24 virtual void activemq::core::ActiveMQConnection::sendPullRequest (const commands::ConsumerInfo * *consumer*, long long *timeout*) throw (exceptions::ActiveMQException) [virtual]

If supported sends a message pull request to the service provider asking for the delivery of a new message. This is used in the case where the service provider has been configured with a zero prefetch or is only capable of delivering messages on a pull basis.

Parameters:

consumer - the ConsumerInfo for the requesting Consumer.

timeout - the time that the client is willing to wait.

6.17.3.25 virtual void activemq::core::ActiveMQConnection::setExceptionListener (cms::ExceptionListener * *listener*) [inline, virtual]

Sets the registered Exception Listener for this connection.

Parameters:

listener pointer to and ExceptionListener

Implements **cms::Connection** (p. 944).

6.17.3.26 virtual void activemq::core::ActiveMQConnection::start () throw (cms::CMSException) [virtual]

Starts or (restarts) a connections delivery of incoming messages.

Exceptions:

CMSException

Implements **cms::Startable** (p. 2413).

6.17.3.27 virtual void activemq::core::ActiveMQConnection::stop () throw (cms::CMSException) [virtual]

Stop the flow of incoming messages.

Exceptions:

CMSException

Implements **cms::Stoppable** (p. 2471).

6.17.3.28 void activemq::core::ActiveMQConnection::syncRequest (Pointer< commands::Command > *command*, unsigned int *timeout* = 0) throw (activemq::exceptions::ActiveMQException)

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

Parameters:

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions:

ConnectorException thrown if an error response was received from the broker, or if any other error occurred.

6.17.3.29 virtual void activemq::core::ActiveMQConnection::transportInterrupted () [virtual]

The **transport** (p. 67) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2625).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnection.h**

6.18 activemq::core::ActiveMQConnectionFactory Class Reference

`#include <src/main/activemq/core/ActiveMQConnectionFactory.h>`Inheritance diagram for activemq::core::ActiveMQConnectionFactory:

Public Member Functions

- **ActiveMQConnectionFactory** ()
- **ActiveMQConnectionFactory** (const std::string &url, const std::string &username="", const std::string &password="")

  Constructor.

- virtual ~**ActiveMQConnectionFactory** ()
- virtual **cms::Connection** * **createConnection** () throw (cms::CMSException)

  Creates a connection with the default user identity.

- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password) throw (cms::CMSException)

  Creates a connection with the specified user identity.

- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password, const std::string &clientId) throw (cms::CMSException)

  Creates a connection with the specified user identity.

- virtual void **setUsername** (const std::string &username)

  Sets the username that should be used when creating a new connection.

- virtual const std::string & **getUsername** () const

  Gets the username that this factory will use when creating a new connection instance.

- virtual void **setPassword** (const std::string &password)

  Sets the password that should be used when creating a new connection.

- virtual const std::string & **getPassword** () const

  Gets the password that this factory will use when creating a new connection instance.

- virtual void **setBrokerURL** (const std::string &brokerURL)

  Sets the Broker URL that should be used when creating a new connection instance.

- virtual const std::string & **getBrokerURL** () const

  Gets the Broker URL that this factory will use when creating a new connection instance.

Static Public Member Functions

- static **cms::Connection** * **createConnection** (const std::string &url, const std::string &username, const std::string &password, const std::string &clientId="") throw (cms::CMSException)

  Creates a connection with the specified user identity.

6.18.1 Constructor & Destructor Documentation

6.18.1.1 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory ()

6.18.1.2 activemq::core::ActiveMQConnectionFactory::ActiveMQConnectionFactory (const std::string & *url*, const std::string & *username* = "", const std::string & *password* = "")

Constructor.

Parameters:

url the URL of the Broker we are connecting to.

username to authenticate with, defaults to ""

password to authenticate with, defaults to ""

6.18.1.3 virtual activemq::core::ActiveMQConnectionFactory::~ActiveMQConnectionFactory () [inline, virtual]

6.18.2 Member Function Documentation

6.18.2.1 static cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *url*, const std::string & *username*, const std::string & *password*, const std::string & *clientId* = "") throw (cms::CMSException) [static]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Parameters:

url the URL of the Broker we are connecting to.

username to authenticate with

password to authenticate with

clientId to assign to connection, defaults to ""

Exceptions:

CMSException.

6.18.2.2 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*, const std::string & *clientId*) throw (cms::CMSException) [virtual]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username to authenticate with

password to authenticate with

clientId to assign to connection if "" then a random cleint Id is created for this connection.

Returns:

a Connection Pointer

Exceptions:

CMSException

Implements **cms::ConnectionFactory** (p. 979).

6.18.2.3 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*) throw (cms::CMSException) [virtual]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username to authenticate with

password to authenticate with

Returns:

a Connection Pointer

Exceptions:

CMSException

Implements **cms::ConnectionFactory** (p. 980).

6.18.2.4 virtual cms::Connection* activemq::core::ActiveMQConnectionFactory::createConnection () throw (cms::CMSException) [virtual]

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the Connection.start method is explicitly called.

Returns:

a Connection Pointer

Exceptions:

CMSException

Implements **cms::ConnectionFactory** (p. 980).

6.18.2.5 virtual const std::string& activemq::core::ActiveMQConnectionFactory::getBrokerURL () const [inline, virtual]

Gets the Broker URL that this factory will use when creating a new connection instance.

Returns:

brokerURL string

6.18.2.6 virtual const std::string& activemq::core::ActiveMQConnectionFactory::getPassword () const [inline, virtual]

Gets the password that this factory will use when creating a new connection instance.

Returns:

password string, "" for default credentials

6.18.2.7 virtual const std::string& activemq::core::ActiveMQConnectionFactory::getUsername () const [inline, virtual]

Gets the username that this factory will use when creating a new connection instance.

Returns:

username string, "" for default credentials

6.18.2.8 virtual void activemq::core::ActiveMQConnectionFactory::setBrokerURL (const std::string & *brokerURL*) [inline, virtual]

Sets the Broker URL that should be used when creating a new connection instance.

Parameters:

brokerURL string

6.18.2.9 virtual void activemq::core::ActiveMQConnectionFactory::setPassword (const std::string & *password*) [inline, virtual]

Sets the password that should be used when creating a new connection.

Parameters:

password string

6.18.2.10 virtual void activemq::core::ActiveMQConnectionFactory::setUsername (const std::string & *username*) [inline, virtual]

Sets the username that should be used when creating a new connection.

Parameters:

username string

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionFactory.h**

6.19 activemq::core::ActiveMQConnectionMetaData Class Reference

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 190) class.

`#include <src/main/activemq/core/ActiveMQConnectionMetaData.h>`Inheritance diagram for activemq::core::ActiveMQConnectionMetaData:

Public Member Functions

- **ActiveMQConnectionMetaData** ()
- virtual ~**ActiveMQConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const throw (cms::CMSException)

  Gets the CMS API version.

- virtual int **getCMSMajorVersion** () const throw (cms::CMSException)

  Gets the CMS major version number.

- virtual int **getCMSMinorVersion** () const throw (cms::CMSException)

  Gets the CMS minor version number.

- virtual std::string **getCMSProviderName** () const throw (cms::CMSException)

  Gets the CMS provider name.

- virtual std::string **getProviderVersion** () const throw (cms::CMSException)

  Gets the CMS provider version.

- virtual int **getProviderMajorVersion** () const throw (cms::CMSException)

  Gets the CMS provider major version number.

- virtual int **getProviderMinorVersion** () const throw (cms::CMSException)

  Gets the CMS provider minor version number.

- virtual std::vector< std::string > **getCMSXPropertyNames** () const throw (cms::CMSException)

  Gets an Vector of the CMSX property names.

6.19.1 Detailed Description

This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 190) class.

Since:

3.0

6.19.2 Constructor & Destructor Documentation

6.19.2.1 activemq::core::ActiveMQConnectionMetaData::ActiveMQConnectionMetaData ()

6.19.2.2 virtual activemq::core::ActiveMQConnectionMetaData::~ActiveMQConnectionMetaData () [virtual]

6.19.3 Member Function Documentation

6.19.3.1 virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMajorVersion () const throw (cms::CMSException) [virtual]

Gets the CMS major version number.

Returns:

the CMS API major version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1016).

6.19.3.2 virtual int activemq::core::ActiveMQConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSException) [virtual]

Gets the CMS minor version number.

Returns:

the CMS API minor version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1016).

6.19.3.3 virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSProviderName () const throw (cms::CMSException) [virtual]

Gets the CMS provider name.

Returns:

the CMS provider name

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1016).

6.19.3.4 virtual std::string activemq::core::ActiveMQConnectionMetaData::getCMSVersion () const throw (cms::CMSException) [virtual]

Gets the CMS API version.

Returns:

the CMS API Version in String form.

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1017).

6.19.3.5 virtual std::vector<std::string> activemq::core::ActiveMQConnectionMetaData::getCMSXPropertyNames () const throw (cms::CMSException) [virtual]

Gets an Vector of the CMSX property names.

Returns:

an Vector of CMSX property names

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1017).

6.19.3.6 virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMajorVersion () const throw (cms::CMSException) [virtual]

Gets the CMS provider major version number.

Returns:

the CMS provider major version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1017).

6.19.3.7 virtual int activemq::core::ActiveMQConnectionMetaData::getProviderMinorVersion () const throw (cms::CMSException) [virtual]

Gets the CMS provider minor version number.

Returns:

the CMS provider minor version number

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1017).

6.19.3.8 virtual std::string activemq::core::ActiveMQConnectionMetaData::getProviderVersion () const throw (cms::CMSException) [virtual]

Gets the CMS provider version.

Returns:

the CMS provider version

Exceptions:

CMSException If the CMS Provider fails to retrieve the metadata due to some internal error.

Implements **cms::ConnectionMetaData** (p. 1018).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionMetaData.h**

6.20 activemq::core::ActiveMQConnectionSupport Class Reference

`#include <src/main/activemq/core/ActiveMQConnectionSupport.h>`Inheritance diagram for activemq::core::ActiveMQConnectionSupport:

Public Member Functions

- **ActiveMQConnectionSupport** (const **Pointer**< **transport::Transport** > &transport, const **Pointer**< **decaf::util::Properties** > &properties)

  *Creates an instance of the **ActiveMQConnectionSupport** (p. 209) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.*

- virtual ~**ActiveMQConnectionSupport** ()
- virtual void **startupTransport** () throw (decaf::lang::Exception)

  Starts the Transport, this should initiate the connection between this client and the Transports endpoint.

- virtual void **shutdownTransport** () throw (decaf::lang::Exception)

  Closes this object and deallocates the appropriate resources.

- const **decaf::util::Properties** & **getProperties** () const

  Gets the Properties object that this Config object was initialized with.

- **transport::Transport** & **getTransport** () const

  Gets the Transport Configured for this Connection.

- bool **isAlwaysSyncSend** () const

  Gets if the Connection should always send things Synchronously.

- void **setAlwaysSyncSend** (bool value)

  Sets if the Connection should always send things Synchronously.

- bool **isUseAsyncSend** () const

  Gets if the useAsyncSend option is set.

- void **setUseAsyncSend** (bool value)

  Sets the useAsyncSend option.

- unsigned int **getSendTimeout** () const

  Gets the assigned send timeout for this Connector.

- void **setSendTimeout** (unsigned int timeout)

  Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

- unsigned int **getCloseTimeout** () const

  Gets the assigned close timeout for this Connector.

- void **setCloseTimeout** (unsigned int timeout)

  Sets the close timeout to use when sending the disconnect request.

- unsigned int **getProducerWindowSize** () const

  Gets the configured producer window size for Producers that are created from this connector.

- void **setProducerWindowSize** (unsigned int windowSize)

  Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

- std::string **getUsername** () const

  Gets the Configured Username.

- void **setUsername** (const std::string &username)

  Sets the Username.

- std::string **getPassword** () const

  Gets the Configured Password.

- void **setPassword** (const std::string &password)

  Sets the Password.

- std::string **getClientId** () const

  Gets the Configured Client Id.

- void **setClientId** (const std::string &clientId)

  Sets the Client Id.

- long long **getNextSessionId** ()

  Get the Next available Session Id.

- long long **getNextTempDestinationId** ()

  Get the Next Temporary Destination Id.

- virtual void **transportResumed** ()

  *The **transport** (p. 67) has resumed after an interruption.*

6.20.1 Constructor & Destructor Documentation

6.20.1.1 activemq::core::ActiveMQConnectionSupport::ActiveMQConnectionSupport (const Pointer< transport::Transport > & *transport*, const Pointer< decaf::util::Properties > & *properties*)

Creates an instance of the **ActiveMQConnectionSupport** (p. 209) class, the most common properties for a connection are pulled from the properties instance or are set to defaults.

Parameters:

transport *(p. 67)* The Transport that this Connection will use for sending Commands to the Broker.

properties The URI configured properties for this connection.

6.20.1.2 virtual activemq::core::ActiveMQConnectionSupport::~ActiveMQConnectionSupport () [virtual]

6.20.2 Member Function Documentation

6.20.2.1 std::string activemq::core::ActiveMQConnectionSupport::getClientId () const [inline]

Gets the Configured Client Id.

Returns:

the clientId.

6.20.2.2 unsigned int activemq::core::ActiveMQConnectionSupport::getCloseTimeout () const [inline]

Gets the assigned close timeout for this Connector.

Returns:

the close timeout configured in the connection uri

6.20.2.3 long long activemq::core::ActiveMQConnectionSupport::getNextSessionId () [inline]

Get the Next available Session Id.

Returns:

the next id in the sequence.

6.20.2.4 long long activemq::core::ActiveMQConnectionSupport::getNextTempDestinationId () [inline]

Get the Next Temporary Destination Id.

Returns:

the next id in the sequence.

6.20.2.5 std::string activemq::core::ActiveMQConnectionSupport::getPassword () const [inline]

Gets the Configured Password.

Returns:

the password.

6.20.2.6 unsigned int activemq::core::ActiveMQConnectionSupport::getProducerWindowSize () const [inline]

Gets the configured producer window size for Producers that are created from this connector. This only applies if there is no send timeout and the producer is able to send asynchronously.

Returns:

size in bytes of messages that this producer can produce before it must block and wait for ProducerAck messages to free resources.

6.20.2.7 const decaf::util::Properties& activemq::core::ActiveMQConnectionSupport::getProperties () const [inline]

Gets the Properties object that this Config object was initialized with.

Returns:

a const reference to the Connection Config.

References decaf::lang::Pointer< T, REFCOUNTER >::get().

6.20.2.8 unsigned int activemq::core::ActiveMQConnectionSupport::getSendTimeout () const [inline]

Gets the assigned send timeout for this Connector.

Returns:

the send timeout configured in the connection uri

6.20.2.9 transport::Transport& activemq::core::ActiveMQConnectionSupport::getTransport () const [inline]

Gets the Transport Configured for this Connection.

Returns:

the configured **transport** (p. 67)

References decaf::lang::Pointer< T, REFCOUNTER >::get().

6.20.2.10 std::string activemq::core::ActiveMQConnectionSupport::getUsername () const [inline]

Gets the Configured Username.

Returns:

the username.

6.20.2.11 bool activemq::core::ActiveMQConnectionSupport::isAlwaysSyncSend () const [inline]

Gets if the Connection should always send things Synchronously.

Returns:

true if sends should always be Synchronous.

6.20.2.12 bool activemq::core::ActiveMQConnectionSupport::isUseAsyncSend () const [inline]

Gets if the useAsyncSend option is set.

Returns:

true if on false if not.

6.20.2.13 void activemq::core::ActiveMQConnectionSupport::setAlwaysSyncSend (bool *value*) [inline]

Sets if the Connection should always send things Synchronously.

Parameters:

value true if sends should always be Synchronous.

6.20.2.14 void activemq::core::ActiveMQConnectionSupport::setClientId (const std::string & *clientId*) [inline]

Sets the Client Id.

Parameters:

clientId - The new clientId value.

6.20.2.15 void activemq::core::ActiveMQConnectionSupport::setCloseTimeout (unsigned int *timeout*) [inline]

Sets the close timeout to use when sending the disconnect request.

Parameters:

timeout - The time to wait for a close message.

6.20.2.16 void activemq::core::ActiveMQConnectionSupport::setPassword (const std::string & *password*) [inline]

Sets the Password.

Parameters:

password - The new password value.

6.20.2.17 void activemq::core::ActiveMQConnectionSupport::setProducerWindowSize (unsigned int *windowSize*) [inline]

Sets the size in Bytes of messages that a producer can send before it is blocked to await a ProducerAck from the broker that frees enough memory to allow another message to be sent.

Parameters:

windowSize - The size in bytes of the Producers memory window.

6.20.2.18 void activemq::core::ActiveMQConnectionSupport::setSendTimeout (unsigned int *timeout*) [inline]

Sets the send timeout to use when sending Message objects, this will cause all messages to be sent using a Synchronous request is non-zero.

Parameters:

timeout - The time to wait for a response.

6.20.2.19 void activemq::core::ActiveMQConnectionSupport::setUseAsyncSend (bool *value*) [inline]

Sets the useAsyncSend option.

Parameters:

value - true to activate, false to disable.

6.20.2.20 void activemq::core::ActiveMQConnectionSupport::setUsername (const std::string & *username*) [inline]

Sets the Username.

Parameters:

username - The new username value.

6.20.2.21 virtual void activemq::core::ActiveMQConnectionSupport::shutdownTransport () throw (decaf::lang::Exception) [virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

Exception

6.20.2.22 virtual void activemq::core::ActiveMQConnectionSupport::startupTransport () throw (decaf::lang::Exception) [virtual]

Starts the Transport, this should initiate the connection between this client and the Transports endpoint.

Exceptions:

Exception

6.20.2.23 virtual void activemq::core::ActiveMQConnectionSupport::transportResumed () [inline, virtual]

The **transport** (p. 67) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2625).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConnectionSupport.h**

6.21 activemq::core::ActiveMQConstants Class Reference

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

`#include <src/main/activemq/core/ActiveMQConstants.h>`

Data Structures

- class **StaticInitializer**

Public Types

- enum **TransactionState** {

  TRANSACTION_STATE_BEGIN = 0, **TRANSACTION_STATE_PREPARE** = 1, **TRANSACTION_STATE_COMMITONEPHASE** = 2, **TRANSACTION_STATE_COMMITTWOPHASE** = 3,

  TRANSACTION_STATE_ROLLBACK = 4, **TRANSACTION_STATE_RECOVER** = 5, **TRANSACTION_STATE_FORGET** = 6, **TRANSACTION_STATE_END** = 7 }
- enum **DestinationActions** { **DESTINATION_ADD_OPERATION** = 0, **DESTINATION_REMOVE_OPERATION** = 1 }
- enum **AckType** {

  ACK_TYPE_DELIVERED = 0, **ACK_TYPE_POISON** = 1, **ACK_TYPE_CONSUMED** = 2, **ACK_TYPE_REDELIVERED** = 3,

  ACK_TYPE_INDIVIDUAL = 4 }
- enum **DestinationOption** {

  CONSUMER_PREFECTCHSIZE, **CUNSUMER_MAXPENDINGMSGLIMIT**, **CONSUMER_NOLOCAL**, **CONSUMER_DISPATCHASYNC**,

  CONSUMER_RETROACTIVE, **CONSUMER_SELECTOR**, **CONSUMER_EXCLUSIVE**, **CONSUMER_PRIORITY**,

  NUM_OPTIONS }

  These values represent the options that can be appended to an Destination name, i.e.
- enum **URIParam** {

  CONNECTION_SENDTIMEOUT, **CONNECTION_PRODUCERWINDOWSIZE**, **CONNECTION_CLOSETIMEOUT**, **CONNECTION_ALWAYSSYNCSEND**,

  CONNECTION_USEASYNCSEND, **PARAM_USERNAME**, **PARAM_PASSWORD**, **PARAM_CLIENTID**,

  NUM_PARAMS }

  These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Static Public Member Functions

- static const std::string & **toString** (const **DestinationOption** option)
- static **DestinationOption toDestinationOption** (const std::string &option)
- static const std::string & **toString** (const **URIParam** option)
- static **URIParam toURIOption** (const std::string &option)

6.21.1 Detailed Description

Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

6.21.2 Member Enumeration Documentation

6.21.2.1 enum activemq::core::ActiveMQConstants::AckType

Enumerator:

ACK_TYPE_DELIVERED
ACK_TYPE_POISON
ACK_TYPE_CONSUMED
ACK_TYPE_REDELIVERED
ACK_TYPE_INDIVIDUAL

6.21.2.2 enum activemq::core::ActiveMQConstants::DestinationActions

Enumerator:

DESTINATION_ADD_OPERATION
DESTINATION_REMOVE_OPERATION

6.21.2.3 enum activemq::core::ActiveMQConstants::DestinationOption

These values represent the options that can be appended to an Destination name, i.e. /topic/-foo?consumer.exclusive=true

Enumerator:

CONSUMER_PREFECTCHSIZE
CUNSUMER_MAXPENDINGMSGLIMIT
CONSUMER_NOLOCAL
CONSUMER_DISPATCHASYNC
CONSUMER_RETROACTIVE
CONSUMER_SELECTOR
CONSUMER_EXCLUSIVE
CONSUMER_PRIORITY
NUM_OPTIONS

6.21.2.4 enum activemq::core::ActiveMQConstants::TransactionState

Enumerator:

TRANSACTION_STATE_BEGIN
TRANSACTION_STATE_PREPARE
TRANSACTION_STATE_COMMITONEPHASE
TRANSACTION_STATE_COMMITTWOPHASE
TRANSACTION_STATE_ROLLBACK
TRANSACTION_STATE_RECOVER
TRANSACTION_STATE_FORGET
TRANSACTION_STATE_END

6.21.2.5 enum activemq::core::ActiveMQConstants::URIParam

These values represent the parameters that can be added to the connection URI that affect the ActiveMQ Core API.

Enumerator:

CONNECTION_SENDTIMEOUT
CONNECTION_PRODUCERWINDOWSIZE
CONNECTION_CLOSETIMEOUT
CONNECTION_ALWAYSSYNCSEND
CONNECTION_USEASYNCSEND
PARAM_USERNAME
PARAM_PASSWORD
PARAM_CLIENTID
NUM_PARAMS

6.21.3 Member Function Documentation

6.21.3.1 static DestinationOption activemq::core::ActiveMQConstants::toDestinationOption (const std::string & *option*) [inline, static]

6.21.3.2 static const std::string& activemq::core::ActiveMQConstants::toString (const URIParam *option*) [inline, static]

6.21.3.3 static const std::string& activemq::core::ActiveMQConstants::toString (const DestinationOption *option*) [inline, static]

6.21.3.4 static URIParam activemq::core::ActiveMQConstants::toURIOption (const std::string & *option*) [inline, static]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

6.22 activemq::core::ActiveMQConsumer Class Reference

`#include <src/main/activemq/core/ActiveMQConsumer.h>`Inheritance diagram for activemq::core::ActiveMQConsumer:

Public Member Functions

- **ActiveMQConsumer** (const **Pointer**< **commands::ConsumerInfo** > &consumerInfo, **ActiveMQSession** *session, const **Pointer**< **ActiveMQTransactionContext** > &transaction)

  Constructor.

- virtual ~**ActiveMQConsumer** ()
- virtual void **start** ()

  Starts the Consumer if not already started and not closed.

- virtual void **stop** ()

  Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again.

- virtual void **close** () throw (cms::CMSException)

  Closes the Consumer.

- virtual **cms::Message** * **receive** () throw (cms::CMSException)

  Synchronously Receive a Message.

- virtual **cms::Message** * **receive** (int millisecs) throw (cms::CMSException)

  Synchronously Receive a Message, time out after defined interval.

- virtual **cms::Message** * **receiveNoWait** () throw (cms::CMSException)

  Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

- virtual void **setMessageListener** (**cms::MessageListener** *listener) throw (cms::CMSException)

  Sets the MessageListener that this class will send notifs on.

- virtual **cms::MessageListener** * **getMessageListener** () const throw (cms::CMSException)

  Gets the MessageListener that this class will send events to.

- virtual std::string **getMessageSelector** () const throw (cms::CMSException)

  Gets this message consumer's message selector expression.

- virtual void **acknowledgeMessage** (const **commands::Message** *message) throw (cms::CMSException)

  Method called to acknowledge the message passed, called from a message when the mode is client ack.

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)

  Called asynchronously by the session to dispatch a message.

- void **acknowledge** () throw (cms::CMSException)

  Method called to acknowledge all messages that have been received so far.

- void **commit** () throw (exceptions::ActiveMQException)

  Called to Commit the current set of messages in this Transaction.

- void **rollback** () throw (exceptions::ActiveMQException)

  Called to Roll back the current set of messages in this Transaction.

- void **doClose** () throw (exceptions::ActiveMQException)

  Performs the actual close operation on this consumer.

- const **commands::ConsumerInfo** & **getConsumerInfo** () const

  Get the Consumer information for this consumer.

- const **commands::ConsumerId** & **getConsumerId** () const

  Get the Consumer Id for this consumer.

- bool **isClosed** () const
- bool **issynchronizationRegistered** () const

  *Has this Consumer Transaction **Synchronization*** (p. 2516) *been added to the transaction.*

- void **setSynchronizationRegistered** (bool value)

  *Sets the **Synchronization*** (p. 2516) *Registered **state*** (p. 65) *of this consumer.*

- bool **iterate** ()

  Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

- void **deliverAcks** () throw (exceptions::ActiveMQException)

  Forces this consumer to send all pending acks to the broker.

- void **clearMessagesInProgress** ()

  Called on a Failover to clear any pending messages.

Protected Member Functions

- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw (cms::CMSException)

  Used by synchronous receive methods to wait for messages to come in.

- void **beforeMessageIsConsumed** (const **Pointer**< **commands::MessageDispatch** > &dispatch)

  Pre-consume processing.

- void **afterMessageIsConsumed** (const **Pointer< commands::MessageDispatch >** &dispatch, bool messageExpired)

  Post-consume processing.

6.22.1 Constructor & Destructor Documentation

6.22.1.1 activemq::core::ActiveMQConsumer::ActiveMQConsumer (const Pointer< commands::ConsumerInfo > & *consumerInfo*, ActiveMQSession * *session*, const Pointer< ActiveMQTransactionContext > & *transaction*)

Constructor.

6.22.1.2 virtual activemq::core::ActiveMQConsumer::~ActiveMQConsumer () [virtual]

6.22.2 Member Function Documentation

6.22.2.1 void activemq::core::ActiveMQConsumer::acknowledge () throw (cms::CMSException)

Method called to acknowledge all messages that have been received so far.

Exceptions:

CMSException

6.22.2.2 virtual void activemq::core::ActiveMQConsumer::acknowledgeMessage (const commands::Message * *message*) throw (cms::CMSException) [virtual]

Method called to acknowledge the message passed, called from a message when the mode is client ack.

Parameters:

message the Message to Acknowledge

Exceptions:

CMSException

Implements **activemq::core::ActiveMQAckHandler** (p. 145).

6.22.2.3 void activemq::core::ActiveMQConsumer::afterMessageIsConsumed (const Pointer< commands::MessageDispatch > & *dispatch*, bool *messageExpired*) [protected]

Post-consume processing.

Parameters:

dispatch - the consumed message

messageExpired - flag indicating if the message has expired.

6.22.2.4 void activemq::core::ActiveMQConsumer::beforeMessageIsConsumed (const Pointer< commands::MessageDispatch > & *dispatch*) [protected]

Pre-consume processing.

Parameters:

dispatch - the message being consumed.

6.22.2.5 void activemq::core::ActiveMQConsumer::clearMessagesInProgress ()

Called on a Failover to clear any pending messages.

6.22.2.6 virtual void activemq::core::ActiveMQConsumer::close () throw (cms::CMSException) [virtual]

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions:

CMSException

Implements **cms::Closeable** (p. 838).

6.22.2.7 void activemq::core::ActiveMQConsumer::commit () throw (exceptions::ActiveMQException)

Called to Commit the current set of messages in this Transaction.

Exceptions:

ActiveMQException

6.22.2.8 void activemq::core::ActiveMQConsumer::deliverAcks () throw (exceptions::ActiveMQException)

Forces this consumer to send all pending acks to the broker.

6.22.2.9 Pointer<MessageDispatch> activemq::core::ActiveMQConsumer::dequeue (long long *timeout*) throw (cms::CMSException) [protected]

Used by synchronous receive methods to wait for messages to come in.

Parameters:

timeout - The maximum number of milliseconds to wait before returning. If -1, it will block until a messages is received or this consumer is closed. If 0, will not block at all. If > 0, will wait at a maximum the specified number of milliseconds before returning.

Returns:

the message, if received within the allotted time. Otherwise NULL.

Exceptions:

InvalidStateException if this consumer is closed upon entering this method.

6.22.2.10 virtual void activemq::core::ActiveMQConsumer::dispatch (const Pointer< MessageDispatch > & *message*) [virtual]

Called asynchronously by the session to dispatch a message.

Parameters:

message dispatch object pointer

Implements **activemq::core::Dispatcher** (p. 1230).

6.22.2.11 void activemq::core::ActiveMQConsumer::doClose () throw (exceptions::ActiveMQException)

Performs the actual close operation on this consumer.

Exceptions:

ActiveMQException

6.22.2.12 const commands::ConsumerId& activemq::core::ActiveMQConsumer::getConsumerId () const [inline]

Get the Consumer Id for this consumer.

Returns:

Reference to a Consumer Id Object

6.22.2.13 const commands::ConsumerInfo& activemq::core::ActiveMQConsumer::getConsumerInfo () const [inline]

Get the Consumer information for this consumer.

Returns:

Reference to a Consumer Info Object

6.22.2.14 virtual cms::MessageListener* activemq::core::ActiveMQConsumer::getMessageListener () const throw (cms::CMSException) [inline, virtual]

Gets the MessageListener that this class will send events to.

Returns:

the currently registered MessageListener interface pointer.

Implements **cms::MessageConsumer** (p. 1796).

6.22.2.15 virtual std::string activemq::core::ActiveMQConsumer::getMessageSelector () const throw (cms::CMSException) [virtual]

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

cms::CMSException (p. ***850***)

Implements **cms::MessageConsumer** (p. 1796).

6.22.2.16 bool activemq::core::ActiveMQConsumer::isClosed () const [inline]

Returns:

if this Consumer has been closed.

6.22.2.17 bool activemq::core::ActiveMQConsumer::issynchronizationRegistered () const [inline]

Has this Consumer Transaction **Synchronization** (p. 2516) been added to the transaction.

Returns:

true if the synchronization has been added.

6.22.2.18 bool activemq::core::ActiveMQConsumer::iterate ()

Deliver any pending messages to the registered MessageListener if there is one, return true if not all dispatched, or false if no listener or all pending messages have been dispatched.

6.22.2.19 virtual cms::Message* activemq::core::ActiveMQConsumer::receive (int *millisecs*) throw (cms::CMSException) [virtual]

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

Parameters:

millisecs the time in milliseconds to wait before returning

Returns:

new message or null on timeout

Exceptions:

CMSException

Implements **cms::MessageConsumer** (p. 1796).

6.22.2.20 virtual cms::Message* activemq::core::ActiveMQConsumer::receive () throw (cms::CMSException) [virtual]

Synchronously Receive a Message.

Returns:

new message

Exceptions:

CMSException

Implements **cms::MessageConsumer** (p. 1797).

6.22.2.21 virtual cms::Message* activemq::core::ActiveMQConsumer::receiveNoWait () throw (cms::CMSException) [virtual]

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message

Exceptions:

CMSException

Implements **cms::MessageConsumer** (p. 1797).

6.22.2.22 void activemq::core::ActiveMQConsumer::rollback () throw (exceptions::ActiveMQException)

Called to Roll back the current set of messages in this Transaction.

Exceptions:

ActiveMQException

6.22.2.23 virtual void activemq::core::ActiveMQConsumer::setMessageListener (cms::MessageListener * *listener*) throw (cms::CMSException) [virtual]

Sets the MessageListener that this class will send notifs on.

Parameters:

listener MessageListener interface pointer

Implements **cms::MessageConsumer** (p. 1797).

6.22.2.24 void activemq::core::ActiveMQConsumer::setSynchronizationRegistered (bool *value*) [inline]

Sets the **Synchronization** (p. 2516) Registered **state** (p. 65) of this consumer.

Parameters:

value - true if registered false otherwise.

6.22.2.25 virtual void activemq::core::ActiveMQConsumer::start () [virtual]

Starts the Consumer if not already started and not closed. A consumer will no deliver messages until started.

6.22.2.26 virtual void activemq::core::ActiveMQConsumer::stop () [virtual]

Stops a Consumer, the Consumer will not deliver any messages that are dispatched to it until it is started again. A Closed Consumer is also a stopped consumer.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConsumer.h**

6.23 activemq::library::ActiveMQCPP Class Reference

`#include <src/main/activemq/library/ActiveMQCPP.h>`

Public Member Functions

- virtual ~**ActiveMQCPP** ()

Static Public Member Functions

- static void **initializeLibrary** ()

  Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf ***library*** *(p. 64).*

- static void **initializeLibrary** (int argc, char **argv)

  Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf ***library*** *(p. 64).*

- static void **shutdownLibrary** ()

  Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point.

Protected Member Functions

- **ActiveMQCPP** ()
- **ActiveMQCPP** (const **ActiveMQCPP** &)
- **ActiveMQCPP** & **operator**= (const **ActiveMQCPP** &)

6.23.1 Constructor & Destructor Documentation

6.23.1.1 activemq::library::ActiveMQCPP::ActiveMQCPP () [inline, protected]

6.23.1.2 activemq::library::ActiveMQCPP::ActiveMQCPP (const ActiveMQCPP &) [protected]

6.23.1.3 virtual activemq::library::ActiveMQCPP::~ActiveMQCPP () [inline, virtual]

6.23.2 Member Function Documentation

6.23.2.1 static void activemq::library::ActiveMQCPP::initializeLibrary (int *argc*, char ** *argv*) [static]

Initialize the ActiveMQ-CPP Library constructs, this method will initialize all the internal Registry objects and initialize the Decaf **library** (p. 64). This method takes the args passed to the main method of process for use is setting system properties and configuring the ActiveMQ-CPP Library.

Parameters:

argc - the count of arguments passed to this Process.

argv - the array of string arguments passed to this process.

Exceptions:

runtime_error if an error occurs while initializing this **library** (p. 64).

6.23.2.2 static void activemq::library::ActiveMQCPP::initializeLibrary () [static]

Initialize the ActiveMQ-CPP Library constructs, this method will init all the internal Registry objects and initialize the Decaf **library** (p. 64).

Exceptions:

runtime_error if an error occurs while initializing this **library** (p. 64).

6.23.2.3 ActiveMQCPP& activemq::library::ActiveMQCPP::operator= (const ActiveMQCPP &) [protected]

6.23.2.4 static void activemq::library::ActiveMQCPP::shutdownLibrary () [static]

Shutdown the ActiveMQ-CPP Library, freeing any resources that could not be freed up to this point. All the user created ActiveMQ-CPP objects and Decaf Library objects should have been destroyed by the time this method is called.

The documentation for this class was generated from the following file:

- src/main/activemq/library/**ActiveMQCPP.h**

6.24 activemq::commands::ActiveMQDestination Class Reference

`#include <src/main/activemq/commands/ActiveMQDestination.h>`Inheritance diagram for activemq::commands::ActiveMQDestination:

Data Structures

- struct **DestinationFilter**

Public Member Functions

- **ActiveMQDestination** ()
- **ActiveMQDestination** (const std::string &**physicalName**)
- virtual ~**ActiveMQDestination** ()
- virtual **ActiveMQDestination** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual const std::string & **getPhysicalName** () const

  Fetch this destination's physical name.

- virtual std::string & **getPhysicalName** ()
- virtual void **setPhysicalName** (const std::string &**physicalName**)

  Set this destination's physical name.

- virtual bool **isAdvisory** () const
- virtual void **setAdvisory** (bool **advisory**)
- virtual bool **isConsumerAdvisory** () const
- virtual bool **isProducerAdvisory** () const
- virtual bool **isConnectionAdvisory** () const
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isOrdered** () const

- virtual void **setOrdered** (bool **ordered**)
- virtual std::string **getOrderedTarget** () const
- virtual void **setOrderedTarget** (const std::string &**orderedTarget**)
- virtual **cms::Destination::DestinationType getDestinationType** () const =0

  Returns the Type of Destination that this object represents.

- virtual bool **isTemporary** () const

  Returns true if a temporary Destination.

- virtual bool **isTopic** () const

  Returns true if a Topic Destination.

- virtual bool **isQueue** () const

  Returns true if a Queue Destination.

- virtual bool **isComposite** () const

  Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

- virtual bool **isWildcard** () const
- const **activemq::util::ActiveMQProperties** & **getOptions** () const
- virtual const **cms::Destination** * **getCMSDestination** () const

Static Public Member Functions

- static std::string **createTemporaryName** (const std::string &clientId)

  Create a temporary name from the clientId.

- static std::string **getClientId** (const **ActiveMQDestination** *destination)

  From a temporary destination find the clientId of the Connection that created it.

- static **Pointer**< **ActiveMQDestination** > **createDestination** (int type, const std::string &name)

  Creates a Destination given the String Name to use and a Type.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQDESTINATION** = 0

Protected Attributes

- bool **exclusive**
- bool **ordered**
- bool **advisory**
- std::string **orderedTarget**
- std::string **physicalName**
- **util::ActiveMQProperties options**

Static Protected Attributes

- static const std::string **ADVISORY_PREFIX**
  prefix for Advisory message destinations
- static const std::string **CONSUMER_ADVISORY_PREFIX**
  prefix for consumer advisory destinations
- static const std::string **PRODUCER_ADVISORY_PREFIX**
  prefix for producer advisory destinations
- static const std::string **CONNECTION_ADVISORY_PREFIX**
  prefix for connection advisory destinations
- static const std::string **DEFAULT_ORDERED_TARGET**
  The default target for ordered destinations.
- static const std::string **TEMP_PREFIX**
- static const std::string **TEMP_POSTFIX**
- static const std::string **COMPOSITE_SEPARATOR**
- static const std::string **QUEUE_QUALIFIED_PREFIX**
- static const std::string **TOPIC_QUALIFIED_PREFIX**
- static const std::string **TEMP_QUEUE_QUALIFED_PREFIX**
- static const std::string **TEMP_TOPIC_QUALIFED_PREFIX**

6.24.1 Constructor & Destructor Documentation

6.24.1.1 activemq::commands::ActiveMQDestination::ActiveMQDestination ()

6.24.1.2 activemq::commands::ActiveMQDestination::ActiveMQDestination (const std::string & *physicalName*)

6.24.1.3 virtual activemq::commands::ActiveMQDestination::~ActiveMQDestination () [inline, virtual]

6.24.2 Member Function Documentation

6.24.2.1 virtual ActiveMQDestination* activemq::commands::ActiveMQDestination::cloneDataStructure () const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 338), **activemq::commands::ActiveMQTempDestination** (p. 400), **activemq::commands::ActiveMQTempQueue** (p. 416), **activemq::commands::ActiveMQTempTopic** (p. 433), and **activemq::commands::ActiveMQTopic** (p. 466).

6.24.2.2 virtual void activemq::commands::ActiveMQDestination::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 338), **activemq::commands::ActiveMQTempDestination** (p. 400), **activemq::commands::ActiveMQTempQueue** (p. 417), **activemq::commands::ActiveMQTempTopic** (p. 434), and **activemq::commands::ActiveMQTopic** (p. 466).

Referenced by activemq::commands::ActiveMQTempDestination::copyDataStructure().

6.24.2.3 static Pointer<ActiveMQDestination> activemq::commands::ActiveMQDestination::createDestination (int *type*, const std::string & *name*) [static]

Creates a Destination given the String Name to use and a Type.

Parameters:

type - The Type of Destination to Create

name - The Name to use in the creation of the Destination

Returns:

Pointer to a new **ActiveMQDestination** (p. 229) instance.

6.24.2.4 static std::string activemq::commands::ActiveMQDestination::createTemporaryName (const std::string & *clientId*) [inline, static]

Create a temporary name from the clientId.

Parameters:

clientId

Returns:

6.24.2.5 virtual bool activemq::commands::ActiveMQDestination::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 417), **activemq::commands::ActiveMQTempTopic** (p. 434), and **activemq::commands::ActiveMQTopic** (p. 467).

Referenced by activemq::commands::ActiveMQTopic::equals(), and activemq::commands::ActiveMQTempDestination::equals().

6.24.2.6 static std::string activemq::commands::ActiveMQDestination::getClientId (const ActiveMQDestination * *destination*) [static]

From a temporary destination find the clientId of the Connection that created it.

Parameters:

destination

Returns:

the clientId or null if not a temporary destination

6.24.2.7 virtual const cms::Destination* activemq::commands::ActiveMQDestination::getCMSDestination () const [inline, virtual]

Returns:

the **cms::Destination** (p. 1190) interface pointer that the objects that derive from this class implement.

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempQueue** (p. 417), **activemq::commands::ActiveMQTempTopic** (p. 434), and **activemq::commands::ActiveMQTopic** (p. 467).

6.24.2.8 virtual unsigned char activemq::commands::ActiveMQDestination::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1176).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), and **activemq::commands::ActiveMQTopic** (p. 467).

6.24.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQDestination::getDestinationType () const [pure virtual]

Returns the Type of Destination that this object represents.

Returns:

int type qualifier.

Implemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), and **activemq::commands::ActiveMQTopic** (p. 467).

6.24.2.10 const activemq::util::ActiveMQProperties& activemq::commands::ActiveMQDestination::getOptions () const [inline]

Returns:

a reference (const) to the options properties for this Destination.

6.24.2.11 virtual std::string activemq::commands::ActiveMQDestination::getOrderedTarget () const [inline, virtual]

Returns:

Returns the orderedTarget.

6.24.2.12 virtual std::string& activemq::commands::ActiveMQDestination::getPhysicalName () [inline, virtual]

6.24.2.13 virtual const std::string& activemq::commands::ActiveMQDestination::getPhysicalName () const [inline, virtual]

Fetch this destination's physical name.

Returns:

const string containing the name

6.24.2.14 virtual bool activemq::commands::ActiveMQDestination::isAdvisory () const [inline, virtual]

Returns:

Returns the advisory.

6.24.2.15 virtual bool activemq::commands::ActiveMQDestination::isComposite () const [inline, virtual]

Returns true if this destination represents a collection of destinations; allowing a set of destinations to be published to or subscribed from in one CMS operation.

Returns:

true if this destination represents a collection of child destinations.

6.24.2.16 virtual bool activemq::commands::ActiveMQDestination::isConnectionAdvisory () const [inline, virtual]

Returns:

true if this is a destination for Connection advisories

6.24.2.17 virtual bool activemq::commands::ActiveMQDestination::isConsumerAdvisory () const [inline, virtual]

Returns:

true if this is a destination for Consumer advisories

6.24.2.18 virtual bool activemq::commands::ActiveMQDestination::isExclusive () const [inline, virtual]

Returns:

Returns the exclusive.

6.24.2.19 virtual bool activemq::commands::ActiveMQDestination::isOrdered () const [inline, virtual]

Returns:

Returns the ordered.

6.24.2.20 virtual bool activemq::commands::ActiveMQDestination::isProducerAdvisory () const [inline, virtual]

Returns:

true if this is a destination for Producer advisories

6.24.2.21 virtual bool activemq::commands::ActiveMQDestination::isQueue () const [inline, virtual]

Returns true if a Queue Destination.

Returns:

true/false

6.24.2.22 virtual bool activemq::commands::ActiveMQDestination::isTemporary () const [inline, virtual]

Returns true if a temporary Destination.

Returns:

true/false

References cms::Destination::TEMPORARY_QUEUE, and cms::Destination::TEMPORARY_TOPIC.

6.24.2.23 virtual bool activemq::commands::ActiveMQDestination::isTopic () const [inline, virtual]

Returns true if a Topic Destination.

Returns:

true/false

References cms::Destination::TEMPORARY_TOPIC, and cms::Destination::TOPIC.

6.24.2.24 virtual bool activemq::commands::ActiveMQDestination::isWildcard () const [inline, virtual]

Returns:

true if the destination matches multiple possible destinations

6.24.2.25 virtual void activemq::commands::ActiveMQDestination::setAdvisory (bool *advisory*) [inline, virtual]

Parameters:

advisory The advisory to set.

6.24.2.26 virtual void activemq::commands::ActiveMQDestination::setExclusive (bool *exclusive*) [inline, virtual]

Parameters:

exclusive The exclusive to set.

6.24.2.27 virtual void activemq::commands::ActiveMQDestination::setOrdered (bool *ordered*) [inline, virtual]

Parameters:

ordered The ordered to set.

6.24.2.28 virtual void activemq::commands::ActiveMQDestination::setOrderedTarget (const std::string & *orderedTarget*) [inline, virtual]

Parameters:

orderedTarget The orderedTarget to set.

6.24.2.29 virtual void activemq::commands::ActiveMQDestination::setPhysicalName (const std::string & *physicalName*) [virtual]

Set this destination's physical name.

Returns:

const string containing the name

6.24.2.30 virtual std::string activemq::commands::ActiveMQDestination::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

Reimplemented in **activemq::commands::ActiveMQQueue** (p. 340), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), and **activemq::commands::ActiveMQTopic** (p. 468).

6.24.3 Field Documentation

6.24.3.1 bool activemq::commands::ActiveMQDestination::advisory [protected]

6.24.3.2 const std::string activemq::commands::ActiveMQDestination::ADVISORY_PREFIX [static, protected]

prefix for Advisory message destinations

6.24.3.3 const std::string activemq::commands::ActiveMQDestination::COMPOSITE_SEPARATOR [static, protected]

6.24.3.4 const std::string activemq::commands::ActiveMQDestination::CONNECTION_ADVISORY_PREFIX [static, protected]

prefix for connection advisory destinations

6.24.3.5 const std::string activemq::commands::ActiveMQDestination::CONSUMER_ADVISORY_PREFIX [static, protected]

prefix for consumer advisory destinations

6.24.3.6 const std::string activemq::commands::ActiveMQDestination::DEFAULT_ORDERED_TARGET [static, protected]

The default target for ordered destinations.

6.24.3.7 bool activemq::commands::ActiveMQDestination::exclusive [protected]

6.24.3.8 const unsigned char activemq::commands::ActiveMQDestination::ID_ACTIVEMQDESTINATION = 0 [static]

6.24.3.9 util::ActiveMQProperties activemq::commands::ActiveMQDestination::options [protected]

6.24.3.10 bool activemq::commands::ActiveMQDestination::ordered [protected]

6.24.3.11 std::string activemq::commands::ActiveMQDestination::orderedTarget [protected]

6.24.3.12 std::string activemq::commands::ActiveMQDestination::physicalName [protected]

6.24.3.13 const std::string activemq::commands::ActiveMQDestination::PRODUCER_ADVISORY_PREFIX [static, protected]

prefix for producer advisory destinations

6.24.3.14 const std::string activemq::commands::ActiveMQDestination::QUEUE_QUALIFIED_PREFIX [static, protected]

6.24.3.15 const std::string activemq::commands::ActiveMQDestination::TEMP_POSTFIX [static, protected]

6.24.3.16 const std::string activemq::commands::ActiveMQDestination::TEMP_PREFIX [static, protected]

6.24.3.17 const std::string activemq::commands::ActiveMQDestination::TEMP_QUEUE_QUALIFED_PREFIX [static, protected]

6.24.3.18 const std::string activemq::commands::ActiveMQDestination::TEMP_TOPIC_QUALIFED_PREFIX [static, protected]

6.24.3.19 const std::string activemq::commands::ActiveMQDestination::TOPIC_QUALIFIED_PREFIX [static, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQDestination.h**

6.25 activemq::wireformat::openwire::marshal::v3::ActiveMQDestination Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 240).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.25.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 240). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.25.2 Constructor & Destructor Documentation

6.25.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::ActiveMQD () [inline]

6.25.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::~ActiveMQ () [inline, virtual]

6.25.3 Member Function Documentation

6.25.3.1 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 342), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 421), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 438), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 470).

6.25.3.2 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 343), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 439), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 471).

6.25.3.3 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 343), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 405), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 422), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 439), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 471).

6.25.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 344), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 405), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 440), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 472).

6.25.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 344), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 406), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 423), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 440), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 472).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQDestinationMarshaller.h**

6.26 activemq::wireformat::openwire::marshal::v1::ActiveMQDestination Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 244).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.26.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 244). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.26.2 Constructor & Destructor Documentation

6.26.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::ActiveMQD () [inline]

6.26.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::~ActiveMQ () [inline, virtual]

6.26.3 Member Function Documentation

6.26.3.1 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 346), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 425), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 442), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 474).

6.26.3.2 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 426), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 443), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 475).

6.26.3.3 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 426), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 443), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 475).

6.26.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 348), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 476).

6.26.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 348), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 410), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 476).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQDestinationMarshaller.h**

6.27 activemq::wireformat::openwire::marshal::v2::ActiveMQDestination Class Reference

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 248).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller:

Public Member Functions

- **ActiveMQDestinationMarshaller** ()
- virtual ~**ActiveMQDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.27.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 248). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.27.2 Constructor & Destructor Documentation

6.27.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::ActiveMQD () [inline]

6.27.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::~ActiveMQ () [inline, virtual]

6.27.3 Member Function Documentation

6.27.3.1 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 350), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 429), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 446), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 478).

6.27.3.2 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 430), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 447), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 479).

6.27.3.3 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 430), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 447), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 479).

6.27.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 352), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 413), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 431), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 448), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 480).

6.27.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 352), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 414), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 431), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 448), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 480).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQDestinationMarshaller.h**

6.28 activemq::exceptions::ActiveMQException Class Reference

`#include <src/main/activemq/exceptions/ActiveMQException.h>`Inheritance diagram for activemq::exceptions::ActiveMQException:

Public Member Functions

- **ActiveMQException** () throw ()

  Default Constructor.

- **ActiveMQException** (const **ActiveMQException** &ex) throw ()

  Copy Constructor.

- **ActiveMQException** (const **decaf::lang::Exception** &ex) throw ()

  Copy Constructor.

- **ActiveMQException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual ~**ActiveMQException** () throw ()
- virtual **ActiveMQException** * **clone** () const

  Clones this exception.

- virtual **cms::CMSException convertToCMSException** () const

  Converts this exception to a new CMSException.

6.28.1 Constructor & Destructor Documentation

6.28.1.1 activemq::exceptions::ActiveMQException::ActiveMQException () throw ()

Default Constructor.

6.28.1.2 activemq::exceptions::ActiveMQException::ActiveMQException (const ActiveMQException & *ex*) throw ()

Copy Constructor.

Parameters:

ex The Exception whose internal data is copied into this instance.

6.28.1.3 activemq::exceptions::ActiveMQException::ActiveMQException (const decaf::lang::Exception & *ex*) throw ()

Copy Constructor.

Parameters:

ex The Exception whose internal data is copied into this instance.

6.28.1.4 activemq::exceptions::ActiveMQException::ActiveMQException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message.

Parameters:

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

msg The message to report.

... The list of primitives that are formatted into the message.

6.28.1.5 virtual activemq::exceptions::ActiveMQException::~ActiveMQException () throw () [virtual]

6.28.2 Member Function Documentation

6.28.2.1 virtual ActiveMQException* activemq::exceptions::ActiveMQException::clone () const [virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this Exception object

Reimplemented from **decaf::lang::Exception** (p. 1271).

Reimplemented in **activemq::exceptions::BrokerException** (p. 590).

6.28.2.2 virtual cms::CMSException activemq::exceptions::ActiveMQException::convertToCMSException () const [virtual]

Converts this exception to a new CMSException.

Returns:

a CMSException with the data from this exception

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**ActiveMQException.h**

6.29 activemq::commands::ActiveMQMapMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQMapMessage.h>`Inheritance diagram for activemq::commands::ActiveMQMapMessage:

Public Member Functions

- **ActiveMQMapMessage** ()
- virtual ~**ActiveMQMapMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual bool **isMarshalAware** () const

  Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

- virtual **ActiveMQMapMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat) throw (decaf::io::IOException)

  Perform any processing needed before an marshal.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::MapMessage** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual std::vector< std::string > **getMapNames** () const throw (cms::CMSException)

  Returns an Enumeration of all the names in the MapMessage object.

- virtual bool **itemExists** (const std::string &name) const throw (cms::CMSException)

  Indicates whether an item exists in this MapMessage object.

- virtual bool **getBoolean** (const std::string &name) const throw (cms::CMSException)

  Returns the Boolean value of the Specified name.

- virtual void **setBoolean** (const std::string &name, bool value) throw (cms::CMSException)

  Sets a boolean value with the specified name into the Map.

- virtual unsigned char **getByte** (const std::string &name) const throw (cms::CMSException)

  Returns the Byte value of the Specified name.

- virtual void **setByte** (const std::string &name, unsigned char value) throw (cms::CMSException)

  Sets a Byte value with the specified name into the Map.

- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const throw (cms::CMSException)

  Returns the Bytes value of the Specified name.

- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value) throw (cms::CMSException)

  Sets a Bytes value with the specified name into the Map.

- virtual char **getChar** (const std::string &name) const throw (cms::CMSException)

  Returns the Char value of the Specified name.

- virtual void **setChar** (const std::string &name, char value) throw (cms::CMSException)

  Sets a Char value with the specified name into the Map.

- virtual double **getDouble** (const std::string &name) const throw (cms::CMSException)

  Returns the Double value of the Specified name.

- virtual void **setDouble** (const std::string &name, double value) throw (cms::CMSException)

  Sets a Double value with the specified name into the Map.

- virtual float **getFloat** (const std::string &name) const throw (cms::CMSException)

  Returns the Float value of the Specified name.

- virtual void **setFloat** (const std::string &name, float value) throw (cms::CMSException)

  Sets a Float value with the specified name into the Map.

- virtual int **getInt** (const std::string &name) const throw (cms::CMSException)

  Returns the Int value of the Specified name.

- virtual void **setInt** (const std::string &name, int value) throw (cms::CMSException)

  Sets a Int value with the specified name into the Map.

- virtual long long **getLong** (const std::string &name) const throw (cms::CMSException)

  Returns the Long value of the Specified name.

- virtual void **setLong** (const std::string &name, long long value) throw (cms::CMSException)

Sets a Long value with the specified name into the Map.

- virtual short **getShort** (const std::string &name) const throw (cms::CMSException)

  Returns the Short value of the Specified name.

- virtual void **setShort** (const std::string &name, short value) throw (cms::CMSException)

  Sets a Short value with the specified name into the Map.

- virtual std::string **getString** (const std::string &name) const throw (cms::CMSException)

  Returns the String value of the Specified name.

- virtual void **setString** (const std::string &name, const std::string &value) throw (cms::CMSException)

  Sets a String value with the specified name into the Map.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMAPMESSAGE** = 25

Protected Member Functions

- **util::PrimitiveMap** & **getMap** () throw (decaf::lang::exceptions::NullPointerException)

  Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

- const **util::PrimitiveMap** & **getMap** () const throw (decaf::lang::exceptions::NullPointerException)
- virtual void **checkMapIsUnmarshalled** () const throw (decaf::lang::exceptions::NullPointerException)

  Performs the unmarshal on the Map if needed, otherwise just returns.

6.29.1 Constructor & Destructor Documentation

6.29.1.1 activemq::commands::ActiveMQMapMessage::ActiveMQMapMessage ()

6.29.1.2 virtual activemq::commands::ActiveMQMapMessage::~ActiveMQMapMessage () [virtual]

6.29.2 Member Function Documentation

6.29.2.1 virtual void activemq::commands::ActiveMQMapMessage::beforeMarshal (wireformat::WireFormat * *wireFormat*) throw (decaf::io::IOException) [virtual]

Perform any processing needed before an marshal.

Parameters:

wireFormat - the OpenWireFormat object in use.

Implements **activemq::wireformat::MarshalAware** (p. 1713).

6.29.2.2 virtual void activemq::commands::ActiveMQMapMessage::checkMapIsUnmarshalled () const throw (decaf::lang::exceptions::NullPointerException) [protected, virtual]

Performs the unmarshal on the Map if needed, otherwise just returns.

6.29.2.3 virtual cms::MapMessage* activemq::commands::ActiveMQMapMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.29.2.4 virtual ActiveMQMapMessage* activemq::commands::ActiveMQMapMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.29.2.5 virtual void activemq::commands::ActiveMQMapMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.29.2.6 virtual bool activemq::commands::ActiveMQMapMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.29.2.7 virtual bool activemq::commands::ActiveMQMapMessage::getBoolean (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Boolean value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1703).

6.29.2.8 virtual unsigned char activemq::commands::ActiveMQMapMessage::getByte (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Byte value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1703).

6.29.2.9 virtual std::vector<unsigned char> activemq::commands::ActiveMQMapMessage::getBytes (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Bytes value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1704).

6.29.2.10 virtual char activemq::commands::ActiveMQMapMessage::getChar (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Char value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1704).

6.29.2.11 virtual unsigned char activemq::commands::ActiveMQMapMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.29.2.12 virtual double activemq::commands::ActiveMQMapMessage::getDouble (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Double value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1704).

6.29.2.13 virtual float activemq::commands::ActiveMQMapMessage::getFloat (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Float value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1705).

6.29.2.14 virtual int activemq::commands::ActiveMQMapMessage::getInt (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Int value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1705).

6.29.2.15 virtual long long activemq::commands::ActiveMQMapMessage::getLong (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Long value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1705).

6.29.2.16 const util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () const throw (decaf::lang::exceptions::NullPointerException) [protected]

6.29.2.17 util::PrimitiveMap& activemq::commands::ActiveMQMapMessage::getMap () throw (decaf::lang::exceptions::NullPointerException) [protected]

Fetches a reference to this objects PrimitiveMap, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns:

reference to a PrimtiveMap.

6.29.2.18 virtual std::vector<std::string> activemq::commands::ActiveMQMapMessage::getMapNames () const throw (cms::CMSException) [virtual]

Returns an Enumeration of all the names in the MapMessage object.

Returns:

STL Vector of String values, each of which is the name of an item in the MapMessage

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1705).

6.29.2.19 virtual short activemq::commands::ActiveMQMapMessage::getShort (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the Short value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1706).

6.29.2.20 virtual std::string activemq::commands::ActiveMQMapMessage::getString (const std::string & *name*) const throw (cms::CMSException) [virtual]

Returns the String value of the Specified name.

Parameters:

name of the value to fetch from the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1706).

6.29.2.21 virtual bool activemq::commands::ActiveMQMapMessage::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

Returns:

true if aware of marshaling

Reimplemented from **activemq::commands::Message** (p. 1746).

6.29.2.22 virtual bool activemq::commands::ActiveMQMapMessage::itemExists (const std::string & *name*) const throw (cms::CMSException) [virtual]

Indicates whether an item exists in this MapMessage object.

Parameters:

name - String name of the Object in question

Returns:

boolean value indicating if the name is in the map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1706).

6.29.2.23 virtual void activemq::commands::ActiveMQMapMessage::setBoolean (const std::string & *name*, bool *value*) throw (cms::CMSException) [virtual]

Sets a boolean value with the specified name into the Map.

Parameters:

name - the name of the boolean

value - the boolean value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1707).

6.29.2.24 virtual void activemq::commands::ActiveMQMapMessage::setByte (const std::string & *name*, unsigned char *value*) throw (cms::CMSException) [virtual]

Sets a Byte value with the specified name into the Map.

Parameters:

name - the name of the Byte

value - the Byte value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1707).

6.29.2.25 virtual void activemq::commands::ActiveMQMapMessage::setBytes (const std::string & *name*, const std::vector< unsigned char > & *value*) throw (cms::CMSException) [virtual]

Sets a Bytes value with the specified name into the Map.

Parameters:

name - the name of the Bytes

value - the Bytes value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1707).

6.29.2.26 virtual void activemq::commands::ActiveMQMapMessage::setChar (const std::string & *name*, char *value*) throw (cms::CMSException) [virtual]

Sets a Char value with the specified name into the Map.

Parameters:

name - the name of the Char

value - the Char value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1708).

6.29.2.27 virtual void activemq::commands::ActiveMQMapMessage::setDouble (const std::string & *name*, double *value*) throw (cms::CMSException) [virtual]

Sets a Double value with the specified name into the Map.

Parameters:

name - the name of the Double

value - the Double value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1708).

6.29.2.28 virtual void activemq::commands::ActiveMQMapMessage::setFloat (const std::string & *name*, float *value*) throw (cms::CMSException) [virtual]

Sets a Float value with the specified name into the Map.

Parameters:

name - the name of the Float

value - the Float value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1708).

6.29.2.29 virtual void activemq::commands::ActiveMQMapMessage::setInt (const std::string & *name*, int *value*) throw (cms::CMSException) [virtual]

Sets a Int value with the specified name into the Map.

Parameters:

name - the name of the Int

value - the Int value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1709).

6.29.2.30 virtual void activemq::commands::ActiveMQMapMessage::setLong (const std::string & *name*, long long *value*) throw (cms::CMSException) [virtual]

Sets a Long value with the specified name into the Map.

Parameters:

name - the name of the Long

value - the Long value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1709).

6.29.2.31 virtual void activemq::commands::ActiveMQMapMessage::setShort (const std::string & *name*, short *value*) throw (cms::CMSException) [virtual]

Sets a Short value with the specified name into the Map.

Parameters:

name - the name of the Short

value - the Short value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1709).

6.29.2.32 virtual void activemq::commands::ActiveMQMapMessage::setString (const std::string & *name*, const std::string & *value*) throw (cms::CMSException) [virtual]

Sets a String value with the specified name into the Map.

Parameters:

name - the name of the String

value - the String value to set in the Map

Exceptions:

CMSException

Implements **cms::MapMessage** (p. 1710).

6.29.2.33 virtual std::string activemq::commands::ActiveMQMapMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.29.3 Field Documentation

6.29.3.1 const unsigned char activemq::commands::ActiveMQMapMessage::ID_ACTIVEMQMAPMESSAGE = 25 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMapMessage.h**

6.30 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 267).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMapMessageMarshaller.h>`Inherit
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual ~**ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.30.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 267). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.30.2 Constructor & Destructor Documentation

6.30.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::ActiveMQ () [inline]

6.30.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::~ActiveM () [inline, virtual]

6.30.3 Member Function Documentation

6.30.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.30.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.30.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.30.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.30.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.30.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.30.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMapMessageMarshaller.h**

6.31 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 271).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMapMessageMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual ~**ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.31.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 271). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.31.2 Constructor & Destructor Documentation

6.31.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::ActiveMQ () [inline]

6.31.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::~ActiveM () [inline, virtual]

6.31.3 Member Function Documentation

6.31.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.31.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.31.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.31.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.31.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.31.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.31.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMapMessageMarshaller.h**

6.32 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 275).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller:

Public Member Functions

- **ActiveMQMapMessageMarshaller** ()
- virtual ~**ActiveMQMapMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.32.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 275). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.32.2 Constructor & Destructor Documentation

6.32.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::ActiveMQ () [inline]

6.32.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::~ActiveM () [inline, virtual]

6.32.3 Member Function Documentation

6.32.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.32.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.32.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.32.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.32.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.32.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.32.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMapMessageMarshaller.h**

6.33 activemq::commands::ActiveMQMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQMessage.h>`Inheritance diagram for activemq::commands::ActiveMQMessage:

Public Member Functions

- **ActiveMQMessage** ()
- virtual ~**ActiveMQMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual **ActiveMQMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::Message** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQMESSAGE** = 23

6.33.1 Constructor & Destructor Documentation

6.33.1.1 activemq::commands::ActiveMQMessage::ActiveMQMessage ()

6.33.1.2 virtual activemq::commands::ActiveMQMessage::~ActiveMQMessage () [inline, virtual]

6.33.2 Member Function Documentation

6.33.2.1 virtual cms::Message* activemq::commands::ActiveMQMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.33.2.2 virtual ActiveMQMessage* activemq::commands::ActiveMQMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.33.2.3 virtual void activemq::commands::ActiveMQMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.33.2.4 virtual bool activemq::commands::ActiveMQMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.33.2.5 virtual unsigned char activemq::commands::ActiveMQMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.33.2.6 virtual std::string activemq::commands::ActiveMQMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.33.3 Field Documentation

6.33.3.1 const unsigned char activemq::commands::ActiveMQMessage::ID_ACTIVEMQMESSAGE = 23 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessage.h**

6.34 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMa Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 282).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.34.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 282). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.34.2 Constructor & Destructor Documentation

6.34.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::ActiveMQMess () [inline]

6.34.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::~ActiveMQMe () [inline, virtual]

6.34.3 Member Function Documentation

6.34.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.34.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.34.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.34.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.34.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.34.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.34.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQMessageMarshaller.h**

6.35 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMa Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 286).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.35.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 286). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.35.2 Constructor & Destructor Documentation

6.35.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::ActiveMQMess () [inline]

6.35.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::~ActiveMQMe () [inline, virtual]

6.35.3 Member Function Documentation

6.35.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.35.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.35.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.35.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.35.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.35.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.35.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQMessageMarshaller.h**

6.36 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMa Class Reference

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 290).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMessageMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller:

Public Member Functions

- **ActiveMQMessageMarshaller** ()
- virtual ~**ActiveMQMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.36.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 290). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.36.2 Constructor & Destructor Documentation

6.36.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::ActiveMQMess () [inline]

6.36.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::~ActiveMQMe () [inline, virtual]

6.36.3 Member Function Documentation

6.36.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.36.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.36.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.36.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.36.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.36.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.36.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQMessageMarshaller.h**

6.37 activemq::commands::ActiveMQMessageTemplate< T > Class Template Reference

`#include <src/main/activemq/commands/ActiveMQMessageTemplate.h>`Inheritance diagram for activemq::commands::ActiveMQMessageTemplate< T >:

Public Member Functions

- **ActiveMQMessageTemplate** ()
- virtual ~**ActiveMQMessageTemplate** ()
- virtual void **acknowledge** () const throw (cms::CMSException)

  Acknowledges all consumed messages of the session of this consumed message.

- virtual void **clearBody** () throw (cms::CMSException)

  Clears out the body of the message.

- virtual void **clearProperties** () throw (cms::CMSException)

  Clears the message properties.

- virtual std::vector< std::string > **getPropertyNames** () const throw (cms::CMSException)

  Retrieves the property names.

- virtual bool **propertyExists** (const std::string &name) const throw (cms::CMSException)

  Indicates whether or not a given property exists.

- virtual bool **getBooleanProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a byte property.

- virtual double **getDoubleProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a double property.

- virtual float **getFloatProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const throw (cms::CMSException)

  Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw (cms::CMSException)

  Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw (cms::CMSException)

  Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value) throw (cms::CMSException)

  Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value) throw (cms::CMSException)

  Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value) throw (cms::CMSException)

  Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value) throw (cms::CMSException)

  Sets a long property.

- virtual void **setShortProperty** (const std::string &name, short value) throw (cms::CMSException)

  Sets a short property.

- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw (cms::CMSException)

  Sets a string property.

- virtual std::string **getCMSCorrelationID** () const throw (cms::CMSException)

  Get the Correlation Id for this message.

- virtual void **setCMSCorrelationID** (const std::string &**correlationId**) throw (cms::CMSException)

  Sets the Correlation Id used by this message.

- virtual int **getCMSDeliveryMode** () const throw (cms::CMSException)

  Gets the DeliveryMode for this message.

- virtual void **setCMSDeliveryMode** (int mode) throw (cms::CMSException)

  Sets the DeliveryMode for this message.

- virtual const **cms::Destination** * **getCMSDestination** () const throw (cms::CMSException)

  Gets the Destination for this ***Message*** *(*p. *1737**), returns a.*

- virtual void **setCMSDestination** (const **cms::Destination** ***destination**) throw (cms::CMSException)

  Sets the Destination for this message.

- virtual long long **getCMSExpiration** () const throw (cms::CMSException)

  Gets the Expiration Time for this ***Message*** *(*p. *1737**).*

- virtual void **setCMSExpiration** (long long expireTime) throw (cms::CMSException)

  Sets the Expiration Time for this message.

- virtual std::string **getCMSMessageID** () const throw (cms::CMSException)

  Gets the CMS ***Message*** *(*p. *1737**) Id for this* ***Message*** *(*p. *1737**).*

- virtual void **setCMSMessageID** (const std::string &id AMQCPP_UNUSED) throw (cms::CMSException)

  Sets the CMS ***Message*** *(*p. *1737**) Id for this message.*

- virtual int **getCMSPriority** () const throw (cms::CMSException)

  Gets the Priority Value for this ***Message*** *(*p. *1737**).*

- virtual void **setCMSPriority** (int **priority**) throw (cms::CMSException)

  Sets the Priority Value for this message.

- virtual bool **getCMSRedelivered** () const throw (cms::CMSException)

  Gets the Redelivered Flag for this ***Message*** *(*p. *1737**).*

- virtual void **setCMSRedelivered** (bool redelivered AMQCPP_UNUSED) throw (cms::CMSException)

  Sets the Redelivered Flag for this message.

- virtual const **cms::Destination** * **getCMSReplyTo** () const throw (cms::CMSException)

  Gets the CMS Reply To Address for this ***Message*** *(*p. *1737**).*

- virtual void **setCMSReplyTo** (const **cms::Destination** ***destination**) throw (cms::CMSException)

  Sets the CMS Reply To Address for this message.

- virtual long long **getCMSTimestamp** () const throw (cms::CMSException)

  Gets the Time Stamp for this ***Message*** *(*p. *1737**).*

- virtual void **setCMSTimestamp** (long long timeStamp) throw (cms::CMSException)
  Sets the Time Stamp for this message.
- virtual std::string **getCMSType** () const throw (cms::CMSException)
  *Gets the CMS **Message** (p. 1737) Type for this **Message** (p. 1737).*
- virtual void **setCMSType** (const std::string &**type**) throw (cms::CMSException)
  *Sets the CMS **Message** (p. 1737) Type for this message.*

Protected Member Functions

- void **checkReadOnlyBody** ()
- void **checkReadOnlyProperties** ()

template<typename T> class activemq::commands::ActiveMQMessageTemplate< T >

6.37.1 Constructor & Destructor Documentation

6.37.1.1 template<typename T> activemq::commands::ActiveMQMessageTemplate< T >::ActiveMQMessageTemplate () [inline]

6.37.1.2 template<typename T> virtual activemq::commands::ActiveMQMessageTemplate< T >::~ActiveMQMessageTemplate () [inline, virtual]

6.37.2 Member Function Documentation

6.37.2.1 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::acknowledge () const throw (cms::CMSException) [inline, virtual]

Acknowledges all consumed messages of the session of this consumed message.

6.37.2.2 template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::checkReadOnlyBody () [inline, protected]

6.37.2.3 template<typename T> void activemq::commands::ActiveMQMessageTemplate< T >::checkReadOnlyProperties () [inline, protected]

6.37.2.4 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearBody () throw (cms::CMSException) [inline, virtual]

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented in **activemq::commands::ActiveMQBytesMessage** (p. 166), and **activemq::commands::ActiveMQStreamMessage** (p. 376).

6.37.2.5 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::clearProperties () throw (cms::CMSException) [inline, virtual]

Clears the message properties. Does not clear the body or header values.

6.37.2.6 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getBooleanProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.7 template<typename T> virtual unsigned char activemq::commands::ActiveMQMessageTemplate< T >::getByteProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.8 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSCorrelationID () const throw (cms::CMSException) [inline, virtual]

Get the Correlation Id for this message.

Returns:

string representation of the correlation Id

Exceptions:

CMSException

6.37.2.9 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getCMSDeliveryMode () const throw (cms::CMSException) [inline, virtual]

Gets the DeliveryMode for this message.

Returns:

DeliveryMode enumerated value.

Exceptions:

CMSException

6.37.2.10 template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSDestination () const throw (cms::CMSException) [inline, virtual]

Gets the Destination for this **Message** (p. 1737), returns a.

Returns:

Destination object

Exceptions:

CMSException

6.37.2.11 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSExpiration () const throw (cms::CMSException) [inline, virtual]

Gets the Expiration Time for this **Message** (p. 1737).

Returns:

time value

Exceptions:

CMSException

6.37.2.12 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSMessageID () const throw (cms::CMSException) [inline, virtual]

Gets the CMS **Message** (p. 1737) Id for this **Message** (p. 1737).

Returns:

time value

Exceptions:

CMSException

6.37.2.13 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getCMSPriority () const throw (cms::CMSException) [inline, virtual]

Gets the Priority Value for this **Message** (p. 1737).

Returns:

priority value

Exceptions:

CMSException

6.37.2.14 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::getCMSRedelivered () const throw (cms::CMSException) [inline, virtual]

Gets the Redelivered Flag for this **Message** (p. 1737).

Returns:

redelivered value

Exceptions:

CMSException

6.37.2.15 template<typename T> virtual const cms::Destination* activemq::commands::ActiveMQMessageTemplate< T >::getCMSReplyTo () const throw (cms::CMSException) [inline, virtual]

Gets the CMS Reply To Address for this **Message** (p. 1737).

Returns:

Reply To Value

Exceptions:

CMSException

6.37.2.16 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getCMSTimestamp () const throw (cms::CMSException) [inline, virtual]

Gets the Time Stamp for this **Message** (p. 1737).

Returns:

time stamp value

Exceptions:

CMSException

6.37.2.17 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getCMSType () const throw (cms::CMSException) [inline, virtual]

Gets the CMS **Message** (p. 1737) Type for this **Message** (p. 1737).

Returns:

type value

Exceptions:

CMSException

6.37.2.18 template<typename T> virtual double activemq::commands::ActiveMQMessageTemplate< T >::getDoubleProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.19 template<typename T> virtual float activemq::commands::ActiveMQMessageTemplate< T >::getFloatProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.20 template<typename T> virtual int activemq::commands::ActiveMQMessageTemplate< T >::getIntProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.21 template<typename T> virtual long long activemq::commands::ActiveMQMessageTemplate< T >::getLongProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.22 template<typename T> virtual std::vector<std::string> activemq::commands::ActiveMQMessageTemplate< T >::getPropertyNames () const throw (cms::CMSException) [inline, virtual]

Retrieves the property names.

Returns:

The complete set of property names currently in this message.

6.37.2.23 template<typename T> virtual short activemq::commands::ActiveMQMessageTemplate< T >::getShortProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.24 template<typename T> virtual std::string activemq::commands::ActiveMQMessageTemplate< T >::getStringProperty (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.37.2.25 template<typename T> virtual bool activemq::commands::ActiveMQMessageTemplate< T >::propertyExists (const std::string & *name*) const throw (cms::CMSException) [inline, virtual]

Indicates whether or not a given property exists.

Parameters:

name The name of the property to look up.

Returns:

True if the property exists in this message.

6.37.2.26 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setBooleanProperty (const std::string & *name*, bool *value*) throw (cms::CMSException) [inline, virtual]

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.27 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setByteProperty (const std::string & *name*, unsigned char *value*) throw (cms::CMSException) [inline, virtual]

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.28 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSCorrelationID (const std::string & *correlationId*) throw (cms::CMSException) [inline, virtual]

Sets the Correlation Id used by this message.

Parameters:

correlationId - String representing the correlation id.

Exceptions:

CMSException

6.37.2.29 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSDeliveryMode (int *mode*) throw (cms::CMSException) [inline, virtual]

Sets the DeliveryMode for this message.

Parameters:

mode - DeliveryMode enumerated value.

Exceptions:

CMSException

6.37.2.30 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSDestination (const cms::Destination * *destination*) throw (cms::CMSException) [inline, virtual]

Sets the Destination for this message.

Parameters:

destination - Destination Object

Exceptions:

CMSException

6.37.2.31 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSExpiration (long long *expireTime*) throw (cms::CMSException) [inline, virtual]

Sets the Expiration Time for this message.

Parameters:

expireTime - time value

Exceptions:

CMSException

6.37.2.32 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSMessageID (const std::string &id *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

Sets the CMS **Message** (p. 1737) Id for this message.

Parameters:

id - time value

Exceptions:

CMSException

6.37.2.33 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSPriority (int *priority*) throw (cms::CMSException) [inline, virtual]

Sets the Priority Value for this message.

Parameters:

priority - priority value for this message

Exceptions:

CMSException

6.37.2.34 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSRedelivered (bool redelivered *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

Sets the Redelivered Flag for this message.

Parameters:

redelivered - boolean redelivered value

Exceptions:

CMSException

6.37.2.35 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSReplyTo (const cms::Destination * *destination*) throw (cms::CMSException) [inline, virtual]

Sets the CMS Reply To Address for this message.

Parameters:

destination Pointer to the CMS Destination that is the Reply-To value.

Exceptions:

CMSException

6.37.2.36 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSTimestamp (long long *timeStamp*) throw (cms::CMSException) [inline, virtual]

Sets the Time Stamp for this message.

Parameters:

timeStamp - integer time stamp value

Exceptions:

CMSException

6.37.2.37 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setCMSType (const std::string & *type*) throw (cms::CMSException) [inline, virtual]

Sets the CMS **Message** (p. 1737) Type for this message.

Parameters:

type - message type value string

Exceptions:

CMSException

6.37.2.38 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setDoubleProperty (const std::string & *name*, double *value*) throw (cms::CMSException) [inline, virtual]

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.39 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setFloatProperty (const std::string & *name*, float *value*) throw (cms::CMSException) [inline, virtual]

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.40 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setIntProperty (const std::string & *name*, int *value*) throw (cms::CMSException) [inline, virtual]

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.41 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setLongProperty (const std::string & *name*, long long *value*) throw (cms::CMSException) [inline, virtual]

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.42 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setShortProperty (const std::string & *name*, short *value*) throw (cms::CMSException) [inline, virtual]

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.37.2.43 template<typename T> virtual void activemq::commands::ActiveMQMessageTemplate< T >::setStringProperty (const std::string & *name*, const std::string & *value*) throw (cms::CMSException) [inline, virtual]

Sets a string property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQMessageTemplate.h**

6.38 activemq::commands::ActiveMQObjectMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQObjectMessage.h>`Inheritance diagram for activemq::commands::ActiveMQObjectMessage:

Public Member Functions

- **ActiveMQObjectMessage** ()
- virtual ~**ActiveMQObjectMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQObjectMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::Message** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQOBJECTMESSAGE** = 26

6.38.1 Constructor & Destructor Documentation

6.38.1.1 activemq::commands::ActiveMQObjectMessage::ActiveMQObjectMessage ()

6.38.1.2 virtual activemq::commands::ActiveMQObjectMessage::~ActiveMQObjectMessage () [inline, virtual]

6.38.2 Member Function Documentation

6.38.2.1 virtual cms::Message* activemq::commands::ActiveMQObjectMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.38.2.2 virtual ActiveMQObjectMessage* activemq::commands::ActiveMQObjectMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.38.2.3 virtual void activemq::commands::ActiveMQObjectMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.38.2.4 virtual bool activemq::commands::ActiveMQObjectMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.38.2.5 virtual unsigned char activemq::commands::ActiveMQObjectMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.38.2.6 virtual std::string activemq::commands::ActiveMQObjectMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.38.3 Field Documentation

6.38.3.1 const unsigned char activemq::commands::ActiveMQObjectMessage::ID_ACTIVEMQOBJECTMESSAGE = 26 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQObjectMessage.h**

6.39 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMess Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 313).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQObjectMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.39.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 313). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.39.2 Constructor & Destructor Documentation

6.39.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::ActiveM () [inline]

6.39.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::~Active () [inline, virtual]

6.39.3 Member Function Documentation

6.39.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.39.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.39.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.39.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.39.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.39.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.39.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQObjectMessageMarshaller.h**

6.40 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMess Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 317).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQObjectMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.40.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 317). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.40.2 Constructor & Destructor Documentation

6.40.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::ActiveM () [inline]

6.40.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::~Active () [inline, virtual]

6.40.3 Member Function Documentation

6.40.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.40.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.40.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.40.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.40.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.40.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.40.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQObjectMessageMarshaller.h**

6.41 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMess Class Reference

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 321).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQObjectMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller:

Public Member Functions

- **ActiveMQObjectMessageMarshaller** ()
- virtual ~**ActiveMQObjectMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.41.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 321). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.41.2 Constructor & Destructor Documentation

6.41.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::ActiveM () [inline]

6.41.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::~Active () [inline, virtual]

6.41.3 Member Function Documentation

6.41.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.41.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.41.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.41.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::looseUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.41.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.41.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightMarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.41.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQObjectMessageMarshaller.h**

6.42 activemq::core::ActiveMQProducer Class Reference

`#include <src/main/activemq/core/ActiveMQProducer.h>`Inheritance diagram for activemq::core::ActiveMQProducer:

Public Member Functions

- **ActiveMQProducer** (const **Pointer**< **commands::ProducerInfo** > &producerInfo, const **Pointer**< **cms::Destination** > &destination, **ActiveMQSession** *session)

  *Constructor, creates an instance of an **ActiveMQProducer** (p. 325).*

- virtual ~**ActiveMQProducer** ()
- virtual void **close** () throw (cms::CMSException)

  Closes the Consumer.

- virtual void **send** (**cms::Message** *message) throw (cms::CMSException)

  Sends the message to the default producer destination.

- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException)

  Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message) throw (cms::CMSException)

  Sends the message to the designated destination.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException)

  Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode) throw (cms::CMSException)

  Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const throw (cms::CMSException)

  Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value) throw (cms::CMSException)

  Sets if Message Ids are disabled for this Producer.

- virtual bool **getDisableMessageID** () const throw (cms::CMSException)

  Gets if Message Ids are disabled for this Producer.

- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSException)

  Sets if Message Time Stamps are disabled for this Producer.

- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSException)

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority) throw (cms::CMSException)

  Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const throw (cms::CMSException)

  Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time) throw (cms::CMSException)

  Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () const throw (cms::CMSException)

  Gets the Time to Live that this producer sends messages with.

- virtual void **setSendTimeout** (long long time) throw (cms::CMSException)

  Sets the Send Timeout that this Producers sends messages with.

- virtual long long **getSendTimeout** () const throw (cms::CMSException)

  Gets the Send Timeout that this producer sends messages with.

- bool **isClosed** () const
- const **commands::ProducerInfo** & **getProducerInfo** () const

  Retries this object ProducerInfo pointer.

- **commands::ProducerId** & **getProducerId** () const

  Retries this object ProducerId or NULL if closed.

- virtual void **onProducerAck** (const **commands::ProducerAck** &ack)

  Handles the work of Processing a ProducerAck Command from the Broker.

6.42.1 Constructor & Destructor Documentation

6.42.1.1 activemq::core::ActiveMQProducer::ActiveMQProducer (const Pointer< commands::ProducerInfo > & *producerInfo*, const Pointer< cms::Destination > & *destination*, ActiveMQSession * *session*)

Constructor, creates an instance of an **ActiveMQProducer** (p. 325).

Parameters:

producerInfo Pointer to a ProducerInfo command which identifies this producer.

destination The assigned Destination this Producer sends to, or null if not set. The Producer does not own the Pointer passed.

session The Session which is the parent of this Producer.

6.42.1.2 virtual activemq::core::ActiveMQProducer::~ActiveMQProducer () [virtual]

6.42.2 Member Function Documentation

6.42.2.1 virtual void activemq::core::ActiveMQProducer::close () throw (cms::CMSException) [virtual]

Closes the Consumer. This will return all allocated resources and purge any outstanding messages. This method will block if there is a call to receive in progress, or a dispatch to a MessageListener in place

Exceptions:

CMSException

Implements **cms::Closeable** (p. 838).

6.42.2.2 virtual int activemq::core::ActiveMQProducer::getDeliveryMode () const throw (cms::CMSException) [inline, virtual]

Gets the delivery mode for this Producer.

Returns:

The DeliveryMode

Implements **cms::MessageProducer** (p. 1879).

6.42.2.3 virtual bool activemq::core::ActiveMQProducer::getDisableMessageID () const throw (cms::CMSException) [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns:

a boolean indicating **state** (p. 65) enable / disable (true / false) for MessageIds.

Implements **cms::MessageProducer** (p. 1880).

6.42.2.4 virtual bool activemq::core::ActiveMQProducer::getDisableMessageTimeStamp () const throw (cms::CMSException) [inline, virtual]

Gets if Message Time Stamps are disabled for this Producer.

Returns:

boolean indicating **state** (p. 65) of enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1880).

6.42.2.5 virtual int activemq::core::ActiveMQProducer::getPriority () const throw (cms::CMSException) [inline, virtual]

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Implements **cms::MessageProducer** (p. 1880).

6.42.2.6 commands::ProducerId& activemq::core::ActiveMQProducer::getProducerId () const [inline]

Retries this object ProducerId or NULL if closed.

Returns:

ProducerId Reference

6.42.2.7 const commands::ProducerInfo& activemq::core::ActiveMQProducer::getProducerInfo () const [inline]

Retries this object ProducerInfo pointer.

Returns:

ProducerInfo Reference

6.42.2.8 virtual long long activemq::core::ActiveMQProducer::getSendTimeout () const throw (cms::CMSException) [inline, virtual]

Gets the Send Timeout that this producer sends messages with.

Returns:

The default send timeout value in milliseconds.

6.42.2.9 virtual long long activemq::core::ActiveMQProducer::getTimeToLive () const throw (cms::CMSException) [inline, virtual]

Gets the Time to Live that this producer sends messages with.

Returns:

The default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 1880).

6.42.2.10 bool activemq::core::ActiveMQProducer::isClosed () const [inline]

Returns:

true if this Producer has been closed.

6.42.2.11 virtual void activemq::core::ActiveMQProducer::onProducerAck (const commands::ProducerAck & *ack*) [virtual]

Handles the work of Processing a ProducerAck Command from the Broker.

Parameters:

ack - The ProducerAck message received from the Broker.

6.42.2.12 virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * *destination*, cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSException) [virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination - a Message Object Pointer

message - a Message Object Pointer

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException

Implements **cms::MessageProducer** (p. 1881).

6.42.2.13 virtual void activemq::core::ActiveMQProducer::send (const cms::Destination * *destination*, cms::Message * *message*) throw (cms::CMSException) [virtual]

Sends the message to the designated destination.

Parameters:

destination The CMS Destination that defines where the message is sent.

message A Message Object Pointer

Exceptions:

CMSException

Implements **cms::MessageProducer** (p. 1881).

6.42.2.14 virtual void activemq::core::ActiveMQProducer::send (cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSException) [virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message A Message Object Pointer

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException

Implements **cms::MessageProducer** (p. 1882).

6.42.2.15 virtual void activemq::core::ActiveMQProducer::send (cms::Message * *message*) throw (cms::CMSException) [virtual]

Sends the message to the default producer destination.

Parameters:

message A Message Object Pointer for the Message to send.

Exceptions:

CMSException

Implements **cms::MessageProducer** (p. 1882).

6.42.2.16 virtual void activemq::core::ActiveMQProducer::setDeliveryMode (int *mode*) throw (cms::CMSException) [inline, virtual]

Sets the delivery mode for this Producer.

Parameters:

mode - The DeliveryMode to use for Message sends.

Implements **cms::MessageProducer** (p. 1882).

6.42.2.17 virtual void activemq::core::ActiveMQProducer::setDisableMessageID (bool *value*) throw (cms::CMSException) [inline, virtual]

Sets if Message Ids are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1883).

6.42.2.18 virtual void activemq::core::ActiveMQProducer::setDisableMessageTimeStamp (bool *value*) throw (cms::CMSException) [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Implements **cms::MessageProducer** (p. 1883).

6.42.2.19 virtual void activemq::core::ActiveMQProducer::setPriority (int *priority*) throw (cms::CMSException) [inline, virtual]

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Implements **cms::MessageProducer** (p. 1883).

6.42.2.20 virtual void activemq::core::ActiveMQProducer::setSendTimeout (long long *time*) throw (cms::CMSException) [inline, virtual]

Sets the Send Timeout that this Producers sends messages with.

Parameters:

time The new default send timeout value in milliseconds.

6.42.2.21 virtual void activemq::core::ActiveMQProducer::setTimeToLive (long long *time*) throw (cms::CMSException) [inline, virtual]

Sets the Time to Live that this Producers sends messages with.

Parameters:

time The new default time to live value in milliseconds.

Implements **cms::MessageProducer** (p. 1883).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQProducer.h**

6.43 activemq::util::ActiveMQProperties Class Reference

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2140) object.

`#include <src/main/activemq/util/ActiveMQProperties.h>`Inheritance diagram for activemq::util::ActiveMQProperties:

Public Member Functions

- virtual ~**ActiveMQProperties** ()
- virtual **decaf::util::Properties** & **getProperties** ()
- virtual const **decaf::util::Properties** & **getProperties** () const
- virtual void **setProperties** (**decaf::util::Properties** &props)
- virtual bool **isEmpty** () const

  Returns true if the properties object is empty.

- virtual const char * **getProperty** (const std::string &name) const

  Looks up the value for the given property.

- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const

  Looks up the value for the given property.

- virtual void **setProperty** (const std::string &name, const std::string &value)

  Sets the value for a given property.

- virtual bool **hasProperty** (const std::string &name) const

  Check to see if the Property exists in the set.

- virtual void **remove** (const std::string &name)

  Removes the property with the given name.

- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const

  Method that serializes the contents of the property map to an arryay.

- virtual void **copy** (const CMSProperties *source)

  Copies the contents of the given properties object to this one.

- virtual CMSProperties * **clone** () const

  Clones this object.

- virtual void **clear** ()

  Clears all properties from the map.

- virtual std::string **toString** () const

  Formats the contents of the Properties Object into a string that can be logged, etc.

6.43.1 Detailed Description

Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2140) object.

Since:

2.0

6.43.2 Constructor & Destructor Documentation

6.43.2.1 virtual activemq::util::ActiveMQProperties::~ActiveMQProperties () [virtual]

6.43.3 Member Function Documentation

6.43.3.1 virtual void activemq::util::ActiveMQProperties::clear () [inline, virtual]

Clears all properties from the map.

Implements **cms::CMSProperties** (p. 854).

6.43.3.2 virtual CMSProperties* activemq::util::ActiveMQProperties::clone () const [virtual]

Clones this object.

Returns:

a replica of this object.

Implements **cms::CMSProperties** (p. 854).

6.43.3.3 virtual void activemq::util::ActiveMQProperties::copy (const CMSProperties * *source*) [virtual]

Copies the contents of the given properties object to this one.

Parameters:

source The source properties object.

6.43.3.4 virtual const decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties () const [inline, virtual]

6.43.3.5 virtual decaf::util::Properties& activemq::util::ActiveMQProperties::getProperties () [inline, virtual]

6.43.3.6 virtual std::string activemq::util::ActiveMQProperties::getProperty (const std::string & *name*, const std::string & *defaultValue*) const [inline, virtual]

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by `name`, if it exists, otherwise the `defaultValue`.

Implements **cms::CMSProperties** (p. 854).

6.43.3.7 virtual const char* activemq::util::ActiveMQProperties::getProperty (const std::string & *name*) const [inline, virtual]

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implements **cms::CMSProperties** (p. 855).

6.43.3.8 virtual bool activemq::util::ActiveMQProperties::hasProperty (const std::string & *name*) const [inline, virtual]

Check to see if the Property exists in the set.

Parameters:

name - property name to check for in this properties set.

Returns:

true if property exists, false otherwise.

Implements **cms::CMSProperties** (p. 855).

6.43.3.9 virtual bool activemq::util::ActiveMQProperties::isEmpty () const [inline, virtual]

Returns true if the properties object is empty.

Returns:

true if empty

Implements **cms::CMSProperties** (p. 855).

6.43.3.10 virtual void activemq::util::ActiveMQProperties::remove (const std::string & *name*) [inline, virtual]

Removes the property with the given name.

Parameters:

name the name of the property to remove.

Implements **cms::CMSProperties** (p. 855).

6.43.3.11 virtual void activemq::util::ActiveMQProperties::setProperties (decaf::util::Properties & *props*) [inline, virtual]

6.43.3.12 virtual void activemq::util::ActiveMQProperties::setProperty (const std::string & *name*, const std::string & *value*) [inline, virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

Implements **cms::CMSProperties** (p. 856).

6.43.3.13 virtual std::vector< std::pair< std::string, std::string > > activemq::util::ActiveMQProperties::toArray () const [inline, virtual]

Method that serializes the contents of the property map to an arryay.

Returns:

list of pairs where the first is the name and the second is the value.

Implements **cms::CMSProperties** (p. 856).

6.43.3.14 virtual std::string activemq::util::ActiveMQProperties::toString () const [inline, virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns:

string value of this object.

Implements **cms::CMSProperties** (p. 856).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**ActiveMQProperties.h**

6.44 activemq::commands::ActiveMQQueue Class Reference

`#include <src/main/activemq/commands/ActiveMQQueue.h>`Inheritance diagram for activemq::commands::ActiveMQQueue:

Public Member Functions

- **ActiveMQQueue** ()
- **ActiveMQQueue** (const std::string &name)
- virtual ~**ActiveMQQueue** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **ActiveMQQueue** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const

  Retrieve the Destination Type for this Destination.

- virtual **cms::Destination** * **clone** () const

  Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)

  Copies the contents of the given Destinastion object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const

  Retrieve any properties that might be part of the destination that was specified.

- virtual std::string **getQueueName** () const throw (cms::CMSException)

  Gets the name of this queue.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQQUEUE** = 100

6.44.1 Constructor & Destructor Documentation

6.44.1.1 activemq::commands::ActiveMQQueue::ActiveMQQueue ()

6.44.1.2 activemq::commands::ActiveMQQueue::ActiveMQQueue (const std::string & *name*)

6.44.1.3 virtual activemq::commands::ActiveMQQueue::~ActiveMQQueue () [inline, virtual]

6.44.2 Member Function Documentation

6.44.2.1 virtual cms::Destination* activemq::commands::ActiveMQQueue::clone () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1191).

6.44.2.2 virtual ActiveMQQueue* activemq::commands::ActiveMQQueue::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 231).

6.44.2.3 virtual void activemq::commands::ActiveMQQueue::copy (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destinastion object to this one.

Parameters:

source The source Destination object.

6.44.2.4 virtual void activemq::commands::ActiveMQQueue::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 232).

6.44.2.5 virtual bool activemq::commands::ActiveMQQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.44.2.6 virtual const cms::Destination* activemq::commands::ActiveMQQueue::getCMSDestination () const [inline, virtual]

Returns:

the **cms::Destination** (p. 1190) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.44.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQQueue::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1191).

6.44.2.8 virtual unsigned char activemq::commands::ActiveMQQueue::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.44.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQQueue::getDestinationType () const [inline, virtual]

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 234).

References cms::Destination::QUEUE.

6.44.2.10 virtual std::string activemq::commands::ActiveMQQueue::getQueueName () const throw (cms::CMSException) [inline, virtual]

Gets the name of this queue.

Returns:

The queue name.

Implements **cms::Queue** (p. 2157).

6.44.2.11 virtual std::string activemq::commands::ActiveMQQueue::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 237).

6.44.3 Field Documentation

6.44.3.1 const unsigned char activemq::commands::ActiveMQQueue::ID_ACTIVEMQQUEUE = 100 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQQueue.h**

6.45 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMars Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 341).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQQueueMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual ~**ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.45.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 341). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.45.2 Constructor & Destructor Documentation

6.45.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::ActiveMQQueue() [inline]

6.45.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::~ActiveMQQueu() [inline, virtual]

6.45.3 Member Function Documentation

6.45.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.45.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::getDataStructureT() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.45.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

6.45.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

6.45.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

6.45.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

6.45.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 243).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQQueueMarshaller.h**

6.46 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMars Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 345).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQQueueMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual ~**ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.46.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 345). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.46.2 Constructor & Destructor Documentation

6.46.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::ActiveMQQueue() [inline]

6.46.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::~ActiveMQQueu() [inline, virtual]

6.46.3 Member Function Documentation

6.46.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.46.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::getDataStructureT() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.46.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

6.46.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

6.46.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

6.46.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

6.46.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 247).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQQueueMarshaller.h**

6.47 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMars Class Reference

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 349).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQQueueMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller:

Public Member Functions

- **ActiveMQQueueMarshaller** ()
- virtual ~**ActiveMQQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.47.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 349). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.47.2 Constructor & Destructor Documentation

6.47.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::ActiveMQQueue() [inline]

6.47.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::~ActiveMQQueu() [inline, virtual]

6.47.3 Member Function Documentation

6.47.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.47.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::getDataStructureT() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.47.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

6.47.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

6.47.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

6.47.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

6.47.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 251).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQQueueMarshaller.h**

6.48 activemq::core::ActiveMQSession Class Reference

`#include <src/main/activemq/core/ActiveMQSession.h>`Inheritance diagram for activemq::core::ActiveMQSession:

Public Member Functions

- **ActiveMQSession** (const **Pointer**< **commands::SessionInfo** > &sessionInfo, **cms::Session::AcknowledgeMode** ackMode, const **decaf::util::Properties** &properties, **ActiveMQConnection** *connection)
- virtual ~**ActiveMQSession** ()
- void **redispatch** (**MessageDispatchChannel** &unconsumedMessages)

  Redispatches the given set of unconsumed messages to the consumers.

- void **start** ()

  Stops asynchronous message delivery.

- void **stop** ()

  Starts asynchronous message delivery.

- bool **isStarted** () const

  *Indicates whether or not the session is currently in the started **state** (p. 65).*

- bool **isAutoAcknowledge** () const
- bool **isDupsOkAcknowledge** () const
- bool **isClientAcknowledge** () const
- void **fire** (const **exceptions::ActiveMQException** &ex)

  Fires the given exception to the exception listener of the connection.

- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)

  Dispatches a message to a particular consumer.

- virtual void **close** () throw (cms::CMSException)

  Closes this session as well as any active child consumers or producers.

- virtual void **commit** () throw (cms::CMSException)

  Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** () throw (cms::CMSException)

  Rollsback all messages done in this transaction and releases any locks currently held.

- virtual void **recover** () throw (cms::CMSException)

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSException)

  Creates a durable subscriber to the specified topic, using a message selector.

- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSException)

  Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSException)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSException)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSException)

  Creates a queue identity given a Queue name.

- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSException)

  Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSException)

  Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSException)

  Creates a TemporaryTopic object.

- virtual **cms::Message** * **createMessage** () throw (cms::CMSException)

  Creates a new Message.

- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSException)

  Creates a BytesMessage.

- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSException)

Creates a BytesMessage and sets the pay-load to the passed value.

- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSException)

  Creates a new StreamMessage.

- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSException)

  Creates a new TextMessage.

- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSException)

  Creates a new TextMessage and set the text to the value given.

- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSException)

  Creates a new MapMessage.

- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSException)

  Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () const throw (cms::CMSException)

  Gets if the Sessions is a Transacted Session.

- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSException)

  Unsubscribes a durable subscription that has been created by a client.

- void **send** (**cms::Message** *message, **ActiveMQProducer** *producer, **util::Usage** *usage) throw (cms::CMSException)

  Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection.

- **cms::ExceptionListener** * **getExceptionListener** ()

  This method gets any registered exception listener of this sessions connection and returns it.

- const **commands::SessionInfo** & **getSessionInfo** () const

  Gets the Session Information object for this session, if the session is closed than this method throws an exception.

- const **commands::SessionId** & **getSessionId** () const

  Gets the Session Id object for this session, if the session is closed than this method throws an exception.

- **ActiveMQConnection** * **getConnection** () const

  *Gets the **ActiveMQConnection** (p. 190) that is associated with this session.*

- void **oneway** (**Pointer< commands::Command >** command) throw (activemq::exceptions::ActiveMQException)

  Sends a oneway message.

- void **syncRequest** (**Pointer< commands::Command >** command, unsigned int timeout=0) throw (activemq::exceptions::ActiveMQException)

Sends a synchronous request and returns the response from the broker.

- void **disposeOf** (**decaf::lang::Pointer< commands::ConsumerId** > id) throw (activemq::exceptions::ActiveMQException)

  Dispose of a Consumer from this session.

- void **disposeOf** (**decaf::lang::Pointer< commands::ProducerId** > id) throw (activemq::exceptions::ActiveMQException)

  Dispose of a Producer from this session.

- void **doStartTransaction** () throw (exceptions::ActiveMQException)

  Starts if not already start a Transaction for this Session.

- void **deliverAcks** ()

  Request that this Session inform all of its consumers to deliver their pending acks.

- void **clearMessagesInProgress** ()

  Request that this Session inform all of its consumers to clear all messages that are currently in progress.

- void **wakeup** ()

  Causes the Session to wakeup its executer and ensure all messages are dispatched.

Friends

- class **ActiveMQSessionExecutor**

6.48.1 Constructor & Destructor Documentation

6.48.1.1 activemq::core::ActiveMQSession::ActiveMQSession (const Pointer< commands::SessionInfo > & *sessionInfo*, cms::Session::AcknowledgeMode *ackMode*, const decaf::util::Properties & *properties*, ActiveMQConnection * *connection*)

6.48.1.2 virtual activemq::core::ActiveMQSession::~ActiveMQSession () [virtual]

6.48.2 Member Function Documentation

6.48.2.1 void activemq::core::ActiveMQSession::clearMessagesInProgress ()

Request that this Session inform all of its consumers to clear all messages that are currently in progress.

6.48.2.2 virtual void activemq::core::ActiveMQSession::close () throw (cms::CMSException) [virtual]

Closes this session as well as any active child consumers or producers.

Exceptions:

CMSException

Implements **cms::Session** (p. 2273).

6.48.2.3 virtual void activemq::core::ActiveMQSession::commit () throw (cms::CMSException) [virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException

Implements **cms::Session** (p. 2274).

6.48.2.4 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * *queue*, const std::string & *selector*) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2274).

6.48.2.5 virtual cms::QueueBrowser* activemq::core::ActiveMQSession::createBrowser (const cms::Queue * *queue*) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2274).

6.48.2.6 virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage (const unsigned char * *bytes*, std::size_t *bytesSize*) throw (cms::CMSException) [virtual]

Creates a BytesMessage and sets the pay-load to the passed value.

Parameters:

bytes - an array of bytes to set in the message

bytesSize - the size of the bytes array, or number of bytes to use

Returns:

a newly created BytesMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2275).

6.48.2.7 virtual cms::BytesMessage* activemq::core::ActiveMQSession::createBytesMessage () throw (cms::CMSException) [virtual]

Creates a BytesMessage.

Returns:

a newly created BytesMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2275).

6.48.2.8 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*, bool *noLocal*) throw (cms::CMSException) [virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions:

CMSException

Implements **cms::Session** (p. 2275).

6.48.2.9 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*) throw (cms::CMSException) [virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination - The Destination that this consumer receiving messages for.

selector - The Message Selector string to use for this destination

Exceptions:

CMSException

Implements **cms::Session** (p. 2276).

6.48.2.10 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createConsumer (const cms::Destination * *destination*) throw (cms::CMSException) [virtual]

Creates a MessageConsumer for the specified destination.

Parameters:

destination - The Destination that this consumer receiving messages for.

Exceptions:

CMSException

Implements **cms::Session** (p. 2276).

6.48.2.11 virtual cms::MessageConsumer* activemq::core::ActiveMQSession::createDurableConsumer (const cms::Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = false) throw (cms::CMSException) [virtual]

Creates a durable subscriber to the specified topic, using a message selector.

Parameters:

destination - the topic to subscribe to

name - The name used to identify the subscription

selector - only messages matching the selector are received

noLocal - if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Exceptions:

CMSException

Implements **cms::Session** (p. 2277).

6.48.2.12 virtual cms::MapMessage* activemq::core::ActiveMQSession::createMapMessage () throw (cms::CMSException) [virtual]

Creates a new MapMessage.

Returns:

a newly created MapMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2277).

6.48.2.13 virtual cms::Message* activemq::core::ActiveMQSession::createMessage () throw (cms::CMSException) [virtual]

Creates a new Message.

Exceptions:

CMSException

Implements **cms::Session** (p. 2278).

6.48.2.14 virtual cms::MessageProducer* activemq::core::ActiveMQSession::createProducer (const cms::Destination * *destination*) throw (cms::CMSException) [virtual]

Creates a MessageProducer to send messages to the specified destination.

Parameters:

destination - the Destination to publish on

Exceptions:

CMSException

Implements **cms::Session** (p. 2278).

6.48.2.15 virtual cms::Queue* activemq::core::ActiveMQSession::createQueue (const std::string & *queueName*) throw (cms::CMSException) [virtual]

Creates a queue identity given a Queue name.

Parameters:

queueName - the name of the new Queue

Exceptions:

CMSException

Implements **cms::Session** (p. 2278).

6.48.2.16 virtual cms::StreamMessage* activemq::core::ActiveMQSession::createStreamMessage () throw (cms::CMSException) [virtual]

Creates a new StreamMessage.

Returns:

a newly created StreamMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2279).

6.48.2.17 virtual cms::TemporaryQueue* activemq::core::ActiveMQSession::createTemporaryQueue () throw (cms::CMSException) [virtual]

Creates a TemporaryQueue object.

Exceptions:

CMSException

Implements **cms::Session** (p. 2279).

6.48.2.18 virtual cms::TemporaryTopic* activemq::core::ActiveMQSession::createTemporaryTopic () throw (cms::CMSException) [virtual]

Creates a TemporaryTopic object.

Exceptions:

CMSException

Implements **cms::Session** (p. 2279).

6.48.2.19 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage (const std::string & *text*) throw (cms::CMSException) [virtual]

Creates a new TextMessage and set the text to the value given.

Parameters:

text - The initial text for the message

Returns:

a newly created TextMessage with the given Text set in the Message body.

Exceptions:

CMSException

Implements **cms::Session** (p. 2279).

6.48.2.20 virtual cms::TextMessage* activemq::core::ActiveMQSession::createTextMessage () throw (cms::CMSException) [virtual]

Creates a new TextMessage.

Returns:

a newly created TextMessage.

Exceptions:

CMSException

Implements **cms::Session** (p. 2280).

6.48.2.21 virtual cms::Topic* activemq::core::ActiveMQSession::createTopic (const std::string & *topicName*) throw (cms::CMSException) [virtual]

Creates a topic identity given a Queue name.

Parameters:

topicName - the name of the new Topic

Exceptions:

CMSException

Implements **cms::Session** (p. 2280).

6.48.2.22 void activemq::core::ActiveMQSession::deliverAcks ()

Request that this Session inform all of its consumers to deliver their pending acks.

6.48.2.23 virtual void activemq::core::ActiveMQSession::dispatch (const Pointer< MessageDispatch > & *message*) [virtual]

Dispatches a message to a particular consumer.

Parameters:

message - the message to be dispatched

Implements **activemq::core::Dispatcher** (p. 1230).

6.48.2.24 void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ProducerId > *id*) throw (activemq::exceptions::ActiveMQException)

Dispose of a Producer from this session. Removes it from the Connection and clean up any resources associated with it.

Parameters:

id - the Id of the Producer to dispose.

Exceptions:

ActiveMQException if an internal error occurs.

6.48.2.25 void activemq::core::ActiveMQSession::disposeOf (decaf::lang::Pointer< commands::ConsumerId > *id*) throw (activemq::exceptions::ActiveMQException)

Dispose of a Consumer from this session. Removes it from the Connection and clean up any resources associated with it.

Parameters:

id - the Id of the Consumer to dispose.

Exceptions:

ActiveMQException if an internal error occurs.

6.48.2.26 void activemq::core::ActiveMQSession::doStartTransaction () throw (exceptions::ActiveMQException)

Starts if not already start a Transaction for this Session. If the session is not a Transacted Session then an exception is thrown. If a transaction is already in progress then this method has no effect.

Exceptions:

ActiveMQException if this is not a Transacted Session.

6.48.2.27 void activemq::core::ActiveMQSession::fire (const exceptions::ActiveMQException & *ex*)

Fires the given exception to the exception listener of the connection.

6.48.2.28 virtual cms::Session::AcknowledgeMode activemq::core::ActiveMQSession::getAcknowledgeMode () const throw (cms::CMSException) [virtual]

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Implements **cms::Session** (p. 2280).

6.48.2.29 ActiveMQConnection* activemq::core::ActiveMQSession::getConnection () const [inline]

Gets the **ActiveMQConnection** (p. 190) that is associated with this session.

6.48.2.30 cms::ExceptionListener* activemq::core::ActiveMQSession::getExceptionListener ()

This method gets any registered exception listener of this sessions connection and returns it. Mainly intended for use by the objects that this session creates so that they can notify the client of **exceptions** (p. 62) that occur in the context of another thread.

Returns:

cms::ExceptionListener (p. 1275) pointer or NULL

6.48.2.31 const commands::SessionId& activemq::core::ActiveMQSession::getSessionId () const [inline]

Gets the Session Id object for this session, if the session is closed than this method throws an exception.

Returns:

SessionId Reference

6.48.2.32 const commands::SessionInfo& activemq::core::ActiveMQSession::getSessionInfo () const [inline]

Gets the Session Information object for this session, if the session is closed than this method throws an exception.

Returns:

SessionInfo Reference

6.48.2.33 bool activemq::core::ActiveMQSession::isAutoAcknowledge () const [inline]

References cms::Session::AUTO_ACKNOWLEDGE.

6.48.2.34 bool activemq::core::ActiveMQSession::isClientAcknowledge () const [inline]

References cms::Session::CLIENT_ACKNOWLEDGE.

6.48.2.35 bool activemq::core::ActiveMQSession::isDupsOkAcknowledge () const [inline]

References cms::Session::DUPS_OK_ACKNOWLEDGE.

6.48.2.36 bool activemq::core::ActiveMQSession::isStarted () const

Indicates whether or not the session is currently in the started **state** (p. 65).

6.48.2.37 virtual bool activemq::core::ActiveMQSession::isTransacted () const throw (cms::CMSException) [virtual]

Gets if the Sessions is a Transacted Session.

Returns:

transacted true - false.

Implements **cms::Session** (p. 2281).

6.48.2.38 void activemq::core::ActiveMQSession::oneway (Pointer< commands::Command > *command*) throw (activemq::exceptions::ActiveMQException)

Sends a oneway message.

Parameters:

command The message to send.

Exceptions:

ActiveMQException if not currently connected, or if the operation fails for any reason.

6.48.2.39 virtual void activemq::core::ActiveMQSession::recover () throw (cms::CMSException) [virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements **cms::Session** (p. 2281).

6.48.2.40 void activemq::core::ActiveMQSession::redispatch (MessageDispatchChannel & *unconsumedMessages*)

Redispatches the given set of unconsumed messages to the consumers.

Parameters:

unconsumedMessages - unconsumed messages to be redelivered.

6.48.2.41 virtual void activemq::core::ActiveMQSession::rollback () throw (cms::CMSException) [virtual]

Rollsback all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException

Implements **cms::Session** (p. 2281).

6.48.2.42 void activemq::core::ActiveMQSession::send (cms::Message * *message*, ActiveMQProducer * *producer*, util::Usage * *usage*) throw (cms::CMSException)

Sends a message from the Producer specified using this session's connection the message will be sent using the best available means depending on the configuration of the connection. Asynchronous sends will be chosen if at all possible.

Parameters:

message The message to send to the broker.

producer The sending Producer

usage Pointer to a Usage tracker which if set will be increased by the size of the given message.

Exceptions:

CMSException

6.48.2.43 void activemq::core::ActiveMQSession::start ()

Stops asynchronous message delivery.

6.48.2.44 void activemq::core::ActiveMQSession::stop ()

Starts asynchronous message delivery.

6.48.2.45 void activemq::core::ActiveMQSession::syncRequest (Pointer< commands::Command > *command*, unsigned int *timeout* = 0) throw (activemq::exceptions::ActiveMQException)

Sends a synchronous request and returns the response from the broker. Converts any error responses into an exception.

Parameters:

command The request command.

timeout The time to wait for a response, default is zero or infinite.

Exceptions:

ActiveMQException thrown if an error response was received from the broker, or if any other error occurred.

6.48.2.46 virtual void activemq::core::ActiveMQSession::unsubscribe (const std::string & *name*) throw (cms::CMSException) [virtual]

Unsubscribes a durable subscription that has been created by a client. This method deletes the **state** (p. 65) being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name the name used to identify this subscription

Exceptions:

CMSException

Implements **cms::Session** (p. 2282).

6.48.2.47 void activemq::core::ActiveMQSession::wakeup ()

Causes the Session to wakeup its executer and ensure all messages are dispatched.

6.48.3 Friends And Related Function Documentation

6.48.3.1 friend class ActiveMQSessionExecutor [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSession.h**

6.49 activemq::core::ActiveMQSessionExecutor Class Reference

Delegate dispatcher for a single session.

`#include <src/main/activemq/core/ActiveMQSessionExecutor.h>`Inheritance diagram for activemq::core::ActiveMQSessionExecutor:

Public Member Functions

- **ActiveMQSessionExecutor** (**ActiveMQSession** *session)

  Creates an un-started executor for the given session.

- virtual ~**ActiveMQSessionExecutor** ()

  *Calls **stop()** (p. 372) then **clear()** (p. 370).*

- virtual void **execute** (const **Pointer**< **MessageDispatch** > &data)

  Executes the dispatch.

- virtual void **executeFirst** (const **Pointer**< **MessageDispatch** > &data)

  Executes the dispatch.

- virtual void **clearMessagesInProgress** ()

  Removes all messages in the Dispatch Channel so that non are delivered.

- virtual bool **hasUncomsumedMessages** () const
- virtual void **wakeup** ()

  wakeup this executer and dispatch any pending messages.

- virtual void **start** ()

  Starts the dispatching.

- virtual void **stop** ()

  Stops dispatching.

- virtual void **close** ()

  Terminates the dispatching thread.

- virtual bool **isRunning** () const
- virtual bool **isEmpty** ()
- virtual void **clear** ()

  Removes all queued messages and destroys them.

- virtual bool **iterate** ()

  *Iterates on the **MessageDispatchChannel** (p. 1805) sending all pending messages to the Consumers they are destined for.*

- std::vector< **Pointer**< **MessageDispatch** > > **getUnconsumedMessages** ()

6.49.1 Detailed Description

Delegate dispatcher for a single session. Contains a thread to provide for asynchronous dispatching.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 activemq::core::ActiveMQSessionExecutor::ActiveMQSessionExecutor (ActiveMQSession * *session*)

Creates an un-started executor for the given session.

6.49.2.2 virtual activemq::core::ActiveMQSessionExecutor::~ActiveMQSessionExecutor () [virtual]

Calls **stop()** (p. 372) then **clear()** (p. 370).

6.49.3 Member Function Documentation

6.49.3.1 virtual void activemq::core::ActiveMQSessionExecutor::clear () [inline, virtual]

Removes all queued messages and destroys them.

6.49.3.2 virtual void activemq::core::ActiveMQSessionExecutor::clearMessagesInProgress () [inline, virtual]

Removes all messages in the Dispatch Channel so that non are delivered.

6.49.3.3 virtual void activemq::core::ActiveMQSessionExecutor::close () [inline, virtual]

Terminates the dispatching thread. Once this is called, the executor is no longer usable.

6.49.3.4 virtual void activemq::core::ActiveMQSessionExecutor::execute (const Pointer< MessageDispatch > & *data*) [virtual]

Executes the dispatch. Adds the given data to the end of the queue.

Parameters:

data - the data to be dispatched.

6.49.3.5 virtual void activemq::core::ActiveMQSessionExecutor::executeFirst (const Pointer< MessageDispatch > & *data*) [virtual]

Executes the dispatch. Adds the given data to the beginning of the queue.

Parameters:

data - the data to be dispatched.

6.49.3.6 std::vector< Pointer<MessageDispatch> > activemq::core::ActiveMQSessionExecutor::getUnconsumedMessages () [inline]

Returns:

a vector containing all the unconsumed messages, this clears the Message Dispatch Channel when called.

6.49.3.7 virtual bool activemq::core::ActiveMQSessionExecutor::hasUncomsumedMessages () const [inline, virtual]

Returns:

true if there are any pending messages in the dispatch channel.

6.49.3.8 virtual bool activemq::core::ActiveMQSessionExecutor::isEmpty () [inline, virtual]

Returns:

true if there are no messages in the Dispatch Channel.

6.49.3.9 virtual bool activemq::core::ActiveMQSessionExecutor::isRunning () const [inline, virtual]

Returns:

true indicates if the executor is started

6.49.3.10 virtual bool activemq::core::ActiveMQSessionExecutor::iterate () [virtual]

Iterates on the **MessageDispatchChannel** (p. 1805) sending all pending messages to the Consumers they are destined for.

Returns:

false if there are no more messages to dispatch.

Implements **activemq::threads::Task** (p. 2520).

6.49.3.11 virtual void activemq::core::ActiveMQSessionExecutor::start () [virtual]

Starts the dispatching.

6.49.3.12 virtual void activemq::core::ActiveMQSessionExecutor::stop () [virtual]

Stops dispatching.

6.49.3.13 virtual void activemq::core::ActiveMQSessionExecutor::wakeup () [virtual]

wakeup this executer and dispatch any pending messages.

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQSessionExecutor.h**

6.50 activemq::commands::ActiveMQStreamMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQStreamMessage.h>`Inheritance diagram for activemq::commands::ActiveMQStreamMessage:

Public Member Functions

- **ActiveMQStreamMessage** ()
- virtual ~**ActiveMQStreamMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQStreamMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual bool **isMarshalAware** () const

  Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat) throw (decaf::io::IOException)

  Perform any processing needed before an marshal.

- virtual **cms::StreamMessage** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual void **clearBody** () throw (cms::CMSException)

  Clears out the body of the message.

- virtual void **reset** () throw (cms::CMSException)

  Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

- virtual bool **readBoolean** () const throw (cms::CMSException)

  Reads a Boolean from the Bytes message stream.

- virtual void **writeBoolean** (bool value) throw (cms::CMSException)

Writes a boolean to the bytes message stream as a 1-byte value.

- virtual unsigned char **readByte** () const throw (cms::CMSException)

  Reads a Byte from the Bytes message stream.

- virtual void **writeByte** (unsigned char value) throw (cms::CMSException)

  Writes a byte to the bytes message stream as a 1-byte value.

- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const throw (cms::CMSException)

  Reads a byte array from the bytes message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value) throw (cms::CMSException)

  Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *&buffer, std::size_t length) const throw (cms::CMSException)

  Reads a portion of the bytes message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length) throw (cms::CMSException)

  Writes a portion of a byte array to the bytes message stream.

- virtual char **readChar** () const throw (cms::CMSException)

  Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value) throw (cms::CMSException)

  Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const throw (cms::CMSException)

  Reads a 32 bit float from the Bytes message stream.

- virtual void **writeFloat** (float value) throw (cms::CMSException)

  Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const throw (cms::CMSException)

  Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value) throw (cms::CMSException)

  Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const throw (cms::CMSException)

  Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value) throw (cms::CMSException)

  Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const throw (cms::CMSException)

Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value) throw (cms::CMSException)

  Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const throw (cms::CMSException)

  Reads a 32 bit signed intger from the Bytes message stream.

- virtual void **writeInt** (int value) throw (cms::CMSException)

  Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const throw (cms::CMSException)

  Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value) throw (cms::CMSException)

  Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const throw (cms::CMSException)

  Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value) throw (cms::CMSException)

  Writes an ASCII String to the Bytes message stream.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQSTREAMMESSAGE** = 27

Protected Member Functions

- void **checkWriteOnlyBody** () const throw (cms::CMSException)

  Throws an exception if in write-only mode.

- **util::PrimitiveList** & **getList** () throw (decaf::lang::exceptions::NullPointerException)

  Fetches a reference to this objects PrimitiveList, if one needs to be created or unmarshaled, this will perform the correct steps.

- const **util::PrimitiveList** & **getList** () const throw (decaf::lang::exceptions::NullPointerException)
- void **checkListIsUnmarshalled** () const throw (decaf::lang::exceptions::NullPointerException)

  Performs the unmarshal on the List if needed, otherwise just returns.

6.50.1 Constructor & Destructor Documentation

6.50.1.1 activemq::commands::ActiveMQStreamMessage::ActiveMQStreamMessage ()

6.50.1.2 virtual activemq::commands::ActiveMQStreamMessage::~ActiveMQStreamMessage () [virtual]

6.50.2 Member Function Documentation

6.50.2.1 virtual void activemq::commands::ActiveMQStreamMessage::beforeMarshal (wireformat::WireFormat * *wireFormat*) throw (decaf::io::IOException) [virtual]

Perform any processing needed before an marshal.

Parameters:

wireFormat - the OpenWireFormat object in use.

Implements **activemq::wireformat::MarshalAware** (p. 1713).

6.50.2.2 void activemq::commands::ActiveMQStreamMessage::checkListIsUnmarshalled () const throw (decaf::lang::exceptions::NullPointerException) [protected]

Performs the unmarshal on the List if needed, otherwise just returns.

6.50.2.3 void activemq::commands::ActiveMQStreamMessage::checkWriteOnlyBody () const throw (cms::CMSException) [protected]

Throws an exception if in write-only mode.

Exceptions:

CMSException.

6.50.2.4 virtual void activemq::commands::ActiveMQStreamMessage::clearBody () throw (cms::CMSException) [inline, virtual]

Clears out the body of the message. This does not clear the headers or properties.

Reimplemented from **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297).

6.50.2.5 virtual cms::StreamMessage* activemq::commands::ActiveMQStreamMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.50.2.6 virtual ActiveMQStreamMessage* activemq::commands::ActiveMQStreamMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.50.2.7 virtual void activemq::commands::ActiveMQStreamMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.50.2.8 virtual bool activemq::commands::ActiveMQStreamMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.50.2.9 virtual unsigned char activemq::commands::ActiveMQStreamMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.50.2.10 const util::PrimitiveList& activemq::commands::ActiveMQStreamMessage::getList () const throw (decaf::lang::exceptions::NullPointerException) [protected]

6.50.2.11 util::PrimitiveList& activemq::commands::ActiveMQStreamMessage::getList () throw (decaf::lang::exceptions::NullPointerException) [protected]

Fetches a reference to this objects PrimitiveList, if one needs to be created or unmarshaled, this will perform the correct steps.

Returns:

reference to a PrimtiveList.

6.50.2.12 virtual bool activemq::commands::ActiveMQStreamMessage::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

Returns:

true if aware of marshaling

Reimplemented from **activemq::commands::Message** (p. 1746).

6.50.2.13 virtual bool activemq::commands::ActiveMQStreamMessage::readBoolean () const throw (cms::CMSException) [virtual]

Reads a Boolean from the Bytes message stream.

Returns:

boolean value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2478).

6.50.2.14 virtual unsigned char activemq::commands::ActiveMQStreamMessage::readByte () const throw (cms::CMSException) [virtual]

Reads a Byte from the Bytes message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2478).

6.50.2.15 virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (unsigned char *& *buffer*, std::size_t *length*) const throw (cms::CMSException) [virtual]

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an IndexOutOfBoundsException is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer - the buffer into which the data is read

length - the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2479).

6.50.2.16 virtual std::size_t activemq::commands::ActiveMQStreamMessage::readBytes (std::vector< unsigned char > & *value*) const throw (cms::CMSException) [virtual]

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value - buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException if an error occurs.

Implements **cms::StreamMessage** (p. 2479).

6.50.2.17 virtual char activemq::commands::ActiveMQStreamMessage::readChar () const throw (cms::CMSException) [virtual]

Reads a Char from the Bytes message stream.

Returns:

char value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2480).

6.50.2.18 virtual double activemq::commands::ActiveMQStreamMessage::readDouble () const throw (cms::CMSException) [virtual]

Reads a 64 bit double from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2480).

6.50.2.19 virtual float activemq::commands::ActiveMQStreamMessage::readFloat () const throw (cms::CMSException) [virtual]

Reads a 32 bit float from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2481).

6.50.2.20 virtual int activemq::commands::ActiveMQStreamMessage::readInt () const throw (cms::CMSException) [virtual]

Reads a 32 bit signed intger from the Bytes message stream.

Returns:

int value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2481).

6.50.2.21 virtual long long activemq::commands::ActiveMQStreamMessage::readLong () const throw (cms::CMSException) [virtual]

Reads a 64 bit long from the Bytes message stream.

Returns:

long long value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2482).

6.50.2.22 virtual short activemq::commands::ActiveMQStreamMessage::readShort () const throw (cms::CMSException) [virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns:

short value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2482).

6.50.2.23 virtual std::string activemq::commands::ActiveMQStreamMessage::readString () const throw (cms::CMSException) [virtual]

Reads an ASCII String from the Bytes message stream.

Returns:

String from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2482).

6.50.2.24 virtual unsigned short activemq::commands::ActiveMQStreamMessage::readUnsignedShort () const throw (cms::CMSException) [virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2483).

6.50.2.25 virtual void activemq::commands::ActiveMQStreamMessage::reset () throw (cms::CMSException) [virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException

6.50.2.26 virtual std::string activemq::commands::ActiveMQStreamMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.50.2.27 virtual void activemq::commands::ActiveMQStreamMessage::writeBoolean (bool *value*) throw (cms::CMSException) [virtual]

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value - boolean to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2483).

6.50.2.28 virtual void activemq::commands::ActiveMQStreamMessage::writeByte (unsigned char *value*) throw (cms::CMSException) [virtual]

Writes a byte to the bytes message stream as a 1-byte value.

Parameters:

value - byte to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2483).

6.50.2.29 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const unsigned char * *value*, std::size_t *offset*, std::size_t *length*) throw (cms::CMSException) [virtual]

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

Parameters:

value - bytes to write to the stream

offset - the initial offset within the byte array

length - the number of bytes to use

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2484).

6.50.2.30 virtual void activemq::commands::ActiveMQStreamMessage::writeBytes (const std::vector< unsigned char > & *value*) throw (cms::CMSException) [virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters:

value - bytes to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2484).

6.50.2.31 virtual void activemq::commands::ActiveMQStreamMessage::writeChar (char *value*) throw (cms::CMSException) [virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters:

value - char to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2485).

6.50.2.32 virtual void activemq::commands::ActiveMQStreamMessage::writeDouble (double *value*) throw (cms::CMSException) [virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters:

value - double to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2485).

6.50.2.33 virtual void activemq::commands::ActiveMQStreamMessage::writeFloat (float *value*) throw (cms::CMSException) [virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters:

value - float to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2485).

6.50.2.34 virtual void activemq::commands::ActiveMQStreamMessage::writeInt (int *value*) throw (cms::CMSException) [virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters:

value - signed int to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2486).

6.50.2.35 virtual void activemq::commands::ActiveMQStreamMessage::writeLong (long long *value*) throw (cms::CMSException) [virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters:

value - signed long long to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2486).

6.50.2.36 virtual void activemq::commands::ActiveMQStreamMessage::writeShort (short *value*) throw (cms::CMSException) [virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters:

value - signed short to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2486).

6.50.2.37 virtual void activemq::commands::ActiveMQStreamMessage::writeString (const std::string & *value*) throw (cms::CMSException) [virtual]

Writes an ASCII String to the Bytes message stream.

Parameters:

value - String to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2487).

6.50.2.38 virtual void activemq::commands::ActiveMQStreamMessage::writeUnsignedShort (unsigned short *value*) throw (cms::CMSException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value - unsigned short to write to the stream

Exceptions:

CMSException

Implements **cms::StreamMessage** (p. 2487).

6.50.3 Field Documentation

6.50.3.1 const unsigned char activemq::commands::ActiveMQStreamMessage::ID_ACTIVEMQSTREAMMESSAGE = 27 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQStreamMessage.h**

6.51 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 387).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQStreamMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.51.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 387). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.51.2 Constructor & Destructor Documentation

6.51.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::ActiveM () [inline]

6.51.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::~Active () [inline, virtual]

6.51.3 Member Function Documentation

6.51.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.51.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.51.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.51.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.51.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.51.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.51.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller::tightUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQStreamMessageMarshaller.h**

6.52 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 391).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQStreamMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.52.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 391). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.52.2 Constructor & Destructor Documentation

6.52.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::ActiveM () [inline]

6.52.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::~Active () [inline, virtual]

6.52.3 Member Function Documentation

6.52.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.52.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.52.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.52.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.52.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.52.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.52.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller::tightUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQStreamMessageMarshaller.h**

6.53 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMes Class Reference

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 395).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQStreamMessageMarshaller.h>`Inh
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller:

Public Member Functions

- **ActiveMQStreamMessageMarshaller** ()
- virtual ~**ActiveMQStreamMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.53.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 395). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.53.2 Constructor & Destructor Documentation

6.53.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::ActiveM () [inline]

6.53.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::~Active () [inline, virtual]

6.53.3 Member Function Documentation

6.53.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.53.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.53.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.53.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.53.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.53.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.53.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller::tightUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQStreamMessageMarshaller.h**

6.54 activemq::commands::ActiveMQTempDestination Class Reference

`#include <src/main/activemq/commands/ActiveMQTempDestination.h>`Inheritance diagram for activemq::commands::ActiveMQTempDestination:

Public Member Functions

- **ActiveMQTempDestination** ()
- **ActiveMQTempDestination** (const std::string &name)
- virtual ~**ActiveMQTempDestination** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **ActiveMQTempDestination** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual void **close** () throw (cms::CMSException)

  Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker.

- void **setConnection** (**core::ActiveMQConnection** *__connection__)

  Sets the Parent Connection that is notified when this destination is destroyed.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPDESTINATION** = 0

Protected Attributes

- **core::ActiveMQConnection** * **connection**

  Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.54.1 Constructor & Destructor Documentation

6.54.1.1 activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination ()

6.54.1.2 activemq::commands::ActiveMQTempDestination::ActiveMQTempDestination (const std::string & *name*)

6.54.1.3 virtual activemq::commands::ActiveMQTempDestination::~ActiveMQTempDestination () [virtual]

6.54.2 Member Function Documentation

6.54.2.1 virtual ActiveMQTempDestination* activemq::commands::ActiveMQTempDestination::cloneDataStructure () const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 231).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 416), and **activemq::commands::ActiveMQTempTopic** (p. 433).

6.54.2.2 virtual void activemq::commands::ActiveMQTempDestination::close () throw (cms::CMSException) [virtual]

Closes down this Destination resulting in a call to dispose of the TempDestination resource at the Broker. This should only be called when the user is certain that they are finished with this destination. The TempDestination is not closed automatically on shutdown. throws **cms::CMSException** (p. 850)

Implements **cms::Closeable** (p. 838).

6.54.2.3 virtual void activemq::commands::ActiveMQTempDestination::copyDataStructure (const DataStructure * *src*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 232).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 417), and **activemq::commands::ActiveMQTempTopic** (p. 434).

References activemq::commands::ActiveMQDestination::copyDataStructure().

6.54.2.4 virtual bool activemq::commands::ActiveMQTempDestination::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 417), and **activemq::commands::ActiveMQTempTopic** (p. 434).

References activemq::commands::ActiveMQDestination::equals().

6.54.2.5 virtual unsigned char activemq::commands::ActiveMQTempDestination::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 418), and **activemq::commands::ActiveMQTempTopic** (p. 435).

6.54.2.6 void activemq::commands::ActiveMQTempDestination::setConnection (core::ActiveMQConnection * *connection*) [inline]

Sets the Parent Connection that is notified when this destination is destroyed.

Parameters:

connection - The parent connection.

6.54.2.7 virtual std::string activemq::commands::ActiveMQTempDestination::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 237).

Reimplemented in **activemq::commands::ActiveMQTempQueue** (p. 418), and **activemq::commands::ActiveMQTempTopic** (p. 435).

6.54.3 Field Documentation

6.54.3.1 core::ActiveMQConnection* activemq::commands::ActiveMQTempDestination::connection [protected]

Connection that we call back on close to allow this resource to be cleaned up correctly at this end and at the Broker End.

6.54.3.2 const unsigned char activemq::commands::ActiveMQTempDestination::ID_ACTIVEMQTEMPDESTINATION = 0 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempDestination.h**

6.55 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDesti Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 403).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>I`
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.55.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 403). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.55.2 Constructor & Destructor Documentation

6.55.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::Active () [inline]

6.55.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::~Acti () [inline, virtual]

6.55.3 Member Function Documentation

6.55.3.1 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 421), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 438).

6.55.3.2 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::looseUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 422), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 439).

6.55.3.3 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 422), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 439).

6.55.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 423), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 440).

6.55.3.5 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller::tightUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 243).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 423), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 440).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempDestinationMarshaller.h**

6.56 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDesti Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 407).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>I`
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.56.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 407). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.56.2 Constructor & Destructor Documentation

6.56.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::Active () [inline]

6.56.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::~Acti () [inline, virtual]

6.56.3 Member Function Documentation

6.56.3.1 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 425), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 442).

6.56.3.2 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::looseUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 426), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 443).

6.56.3.3 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 426), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 443).

6.56.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444).

6.56.3.5 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller::tightUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 247).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempDestinationMarshaller.h**

6.57 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDesti Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 411).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>I`
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller:

Public Member Functions

- **ActiveMQTempDestinationMarshaller** ()
- virtual ~**ActiveMQTempDestinationMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.57.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 411). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.57.2 Constructor & Destructor Documentation

6.57.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::Active () [inline]

6.57.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::~Acti () [inline, virtual]

6.57.3 Member Function Documentation

6.57.3.1 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 429), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 446).

6.57.3.2 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::looseUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 430), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 447).

6.57.3.3 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 430), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 447).

6.57.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightMa (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 431), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 448).

6.57.3.5 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller::tightUn (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 251).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 431), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 448).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempDestinationMarshaller.h**

6.58 activemq::commands::ActiveMQTempQueue Class Reference

`#include <src/main/activemq/commands/ActiveMQTempQueue.h>`Inheritance diagram for activemq::commands::ActiveMQTempQueue:

Public Member Functions

- **ActiveMQTempQueue** ()
- **ActiveMQTempQueue** (const std::string &name)
- virtual ~**ActiveMQTempQueue** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **ActiveMQTempQueue** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  Converts the Destination Name into a String.

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const

  Retrieve the Destination Type for this Destination.

- virtual **cms::Destination** * **clone** () const

  Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)

  Copies the contents of the given Destinastion object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const

  Retrieve any properties that might be part of the destination that was specified.

- virtual std::string **getQueueName** () const throw (cms::CMSException)

  Gets the name of this queue.

- virtual void **destroy** () throw (cms::CMSException)

  Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPQUEUE** = 102

6.58.1 Constructor & Destructor Documentation

6.58.1.1 activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue ()

6.58.1.2 activemq::commands::ActiveMQTempQueue::ActiveMQTempQueue (const std::string & *name*)

6.58.1.3 virtual activemq::commands::ActiveMQTempQueue::~ActiveMQTempQueue () [virtual]

6.58.2 Member Function Documentation

6.58.2.1 virtual cms::Destination* activemq::commands::ActiveMQTempQueue::clone () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1191).

6.58.2.2 virtual ActiveMQTempQueue* activemq::commands::ActiveMQTempQueue::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 400).

6.58.2.3 virtual void activemq::commands::ActiveMQTempQueue::copy (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destinastion object to this one.

Parameters:

source The source Destination object.

6.58.2.4 virtual void activemq::commands::ActiveMQTempQueue::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 400).

6.58.2.5 virtual void activemq::commands::ActiveMQTempQueue::destroy () throw (cms::CMSException) [inline, virtual]

Destroy's the Temp Destination at the Broker.

Exceptions:

CMSException

Implements **cms::TemporaryQueue** (p. 2538).

References AMQ_CATCH_ALL_THROW_CMSEXCEPTION.

6.58.2.6 virtual bool activemq::commands::ActiveMQTempQueue::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.58.2.7 virtual const cms::Destination* activemq::commands::ActiveMQTempQueue::getCMSDestination () const [inline, virtual]

Returns:

the **cms::Destination** (p. 1190) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.58.2.8 virtual const cms::CMSProperties& activemq::commands::ActiveMQTempQueue::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1191).

6.58.2.9 virtual unsigned char activemq::commands::ActiveMQTempQueue::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.58.2.10 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempQueue::getDestinationType () const [inline, virtual]

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 234).

References cms::Destination::TEMPORARY_QUEUE.

6.58.2.11 virtual std::string activemq::commands::ActiveMQTempQueue::getQueueName () const throw (cms::CMSException) [inline, virtual]

Gets the name of this queue.

Returns:

The queue name.

Implements **cms::TemporaryQueue** (p. 2539).

6.58.2.12 virtual std::string activemq::commands::ActiveMQTempQueue::toString () const [virtual]

Converts the Destination Name into a String.

Returns:

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.58.3 Field Documentation

6.58.3.1 const unsigned char activemq::commands::ActiveMQTempQueue::ID_ACTIVEMQTEMPQUEUE = 102 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempQueue.h**

6.59 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueu Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 420).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempQueueMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual ~**ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.59.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 420). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.59.2 Constructor & Destructor Documentation

6.59.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::ActiveMQT () [inline]

6.59.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::~ActiveMQ () [inline, virtual]

6.59.3 Member Function Documentation

6.59.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.59.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.59.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 404).

6.59.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 404).

6.59.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 405).

6.59.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 405).

6.59.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 406).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempQueueMarshaller.h**

6.60 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueu Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 424).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempQueueMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual ~**ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.60.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 424). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.60.2 Constructor & Destructor Documentation

6.60.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::ActiveMQT () [inline]

6.60.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::~ActiveMQ () [inline, virtual]

6.60.3 Member Function Documentation

6.60.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.60.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.60.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 408).

6.60.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 408).

6.60.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 409).

6.60.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 409).

6.60.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 410).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempQueueMarshaller.h**

6.61 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueu Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 428).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempQueueMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller:

Public Member Functions

- **ActiveMQTempQueueMarshaller** ()
- virtual ~**ActiveMQTempQueueMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.61.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 428). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.61.2 Constructor & Destructor Documentation

6.61.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::ActiveMQT () [inline]

6.61.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::~ActiveMQ () [inline, virtual]

6.61.3 Member Function Documentation

6.61.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.61.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.61.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 412).

6.61.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 412).

6.61.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 413).

6.61.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 413).

6.61.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 414).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempQueueMarshaller.h**

6.62 activemq::commands::ActiveMQTempTopic Class Reference

`#include <src/main/activemq/commands/ActiveMQTempTopic.h>`Inheritance diagram for activemq::commands::ActiveMQTempTopic:

Public Member Functions

- **ActiveMQTempTopic** ()
- **ActiveMQTempTopic** (const std::string &name)
- virtual ~**ActiveMQTempTopic** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **ActiveMQTempTopic** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  Converts the Destination Name into a String.

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const

  Retrieve the Destination Type for this Destination.

- virtual **cms::Destination** * **clone** () const

  Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)

  Copies the contents of the given Destinastion object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const

  Retrieve any properties that might be part of the destination that was specified.

- virtual std::string **getTopicName** () const throw (cms::CMSException)

  Gets the name of this topic.

- virtual void **destroy** () throw (cms::CMSException)

  Destroy's the Temp Destination at the Broker.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEMPTOPIC** = 103

6.62.1 Constructor & Destructor Documentation

6.62.1.1 activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic ()

6.62.1.2 activemq::commands::ActiveMQTempTopic::ActiveMQTempTopic (const std::string & *name*)

6.62.1.3 virtual activemq::commands::ActiveMQTempTopic::~ActiveMQTempTopic () [virtual]

6.62.2 Member Function Documentation

6.62.2.1 virtual cms::Destination* activemq::commands::ActiveMQTempTopic::clone () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1191).

6.62.2.2 virtual ActiveMQTempTopic* activemq::commands::ActiveMQTempTopic::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 400).

6.62.2.3 virtual void activemq::commands::ActiveMQTempTopic::copy (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destinastion object to this one.

Parameters:

source The source Destination object.

6.62.2.4 virtual void activemq::commands::ActiveMQTempTopic::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 400).

6.62.2.5 virtual void activemq::commands::ActiveMQTempTopic::destroy () throw (cms::CMSException) [inline, virtual]

Destroy's the Temp Destination at the Broker.

Exceptions:

CMSException

Implements **cms::TemporaryTopic** (p. 2540).

References AMQ_CATCH_ALL_THROW_CMSEXCEPTION.

6.62.2.6 virtual bool activemq::commands::ActiveMQTempTopic::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.62.2.7 virtual const cms::Destination* activemq::commands::ActiveMQTempTopic::getCMSDestination () const [inline, virtual]

Returns:

the **cms::Destination** (p. 1190) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.62.2.8 virtual const cms::CMSProperties& activemq::commands::ActiveMQTempTopic::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1191).

6.62.2.9 virtual unsigned char activemq::commands::ActiveMQTempTopic::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.62.2.10 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTempTopic::getDestinationType () const [inline, virtual]

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 234).

References cms::Destination::TEMPORARY_TOPIC.

6.62.2.11 virtual std::string activemq::commands::ActiveMQTempTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]

Gets the name of this topic.

Returns:

The topic name.

Implements **cms::TemporaryTopic** (p. 2541).

6.62.2.12 virtual std::string activemq::commands::ActiveMQTempTopic::toString () const [virtual]

Converts the Destination Name into a String.

Returns:

string name

Reimplemented from **activemq::commands::ActiveMQTempDestination** (p. 401).

6.62.3 Field Documentation

6.62.3.1 const unsigned char activemq::commands::ActiveMQTempTopic::ID_ACTIVEMQTEMPTOPIC = 103 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTempTopic.h**

6.63 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 437).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTempTopicMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.63.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 437). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.63.2 Constructor & Destructor Documentation

6.63.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::ActiveMQTe () [inline]

6.63.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::~ActiveMQ () [inline, virtual]

6.63.3 Member Function Documentation

6.63.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.63.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.63.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 404).

6.63.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 404).

6.63.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 405).

6.63.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 405).

6.63.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshalle** (p. 406).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTempTopicMarshaller.h**

6.64 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 441).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.64.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 441). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.64.2 Constructor & Destructor Documentation

6.64.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::ActiveMQTe () [inline]

6.64.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::~ActiveMQ () [inline, virtual]

6.64.3 Member Function Documentation

6.64.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.64.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.64.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 408).

6.64.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 408).

6.64.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 409).

6.64.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 409).

6.64.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshalle** (p. 410).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTempTopicMarshaller.h**

6.65 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopic Class Reference

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 445).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTempTopicMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller:

Public Member Functions

- **ActiveMQTempTopicMarshaller** ()
- virtual ~**ActiveMQTempTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.65.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 445). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.65.2 Constructor & Destructor Documentation

6.65.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::ActiveMQTe () [inline]

6.65.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::~ActiveMQ () [inline, virtual]

6.65.3 Member Function Documentation

6.65.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.65.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.65.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 412).

6.65.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 412).

6.65.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 413).

6.65.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 413).

6.65.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshalle** (p. 414).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTempTopicMarshaller.h**

6.66 activemq::commands::ActiveMQTextMessage Class Reference

`#include <src/main/activemq/commands/ActiveMQTextMessage.h>`Inheritance diagram for activemq::commands::ActiveMQTextMessage:

Public Member Functions

- **ActiveMQTextMessage** ()
- virtual ~**ActiveMQTextMessage** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ActiveMQTextMessage** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual **cms::TextMessage** * **clone** () const

  Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual std::string **getText** () const throw (cms::CMSException)

  Gets the message character buffer.

- virtual void **setText** (const char *msg) throw (cms::CMSException)

  Sets the message contents, does not take ownership of the passed char, but copies it instead.*

- virtual void **setText** (const std::string &msg) throw (cms::CMSException)

  Sets the message contents.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTEXTMESSAGE** = 28

6.66.1 Constructor & Destructor Documentation

6.66.1.1 activemq::commands::ActiveMQTextMessage::ActiveMQTextMessage ()

6.66.1.2 virtual activemq::commands::ActiveMQTextMessage::~ActiveMQTextMessage () [virtual]

6.66.2 Member Function Documentation

6.66.2.1 virtual cms::TextMessage* activemq::commands::ActiveMQTextMessage::clone () const [inline, virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implements **cms::Message** (p. 1758).

6.66.2.2 virtual ActiveMQTextMessage* activemq::commands::ActiveMQTextMessage::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Message** (p. 1741).

6.66.2.3 virtual void activemq::commands::ActiveMQTextMessage::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::Message** (p. 1742).

6.66.2.4 virtual bool activemq::commands::ActiveMQTextMessage::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Message** (p. 1742).

6.66.2.5 virtual unsigned char activemq::commands::ActiveMQTextMessage::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Message** (p. 1743).

6.66.2.6 virtual std::string activemq::commands::ActiveMQTextMessage::getText () const throw (cms::CMSException) [virtual]

Gets the message character buffer.

Returns:

The message character buffer.

Implements **cms::TextMessage** (p. 2542).

6.66.2.7 virtual void activemq::commands::ActiveMQTextMessage::setText (const std::string & *msg*) throw (cms::CMSException) [virtual]

Sets the message contents.

Parameters:

msg The message buffer.

Implements **cms::TextMessage** (p. 2543).

6.66.2.8 virtual void activemq::commands::ActiveMQTextMessage::setText (const char * *msg*) throw (cms::CMSException) [virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters:

msg The message buffer.

Implements **cms::TextMessage** (p. 2543).

6.66.2.9 virtual std::string activemq::commands::ActiveMQTextMessage::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Message** (p. 1749).

6.66.3 Field Documentation

6.66.3.1 const unsigned char activemq::commands::ActiveMQTextMessage::ID_ACTIVEMQTEXTMESSAGE = 28 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTextMessage.h**

6.67 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 453).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTextMessageMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual ~**ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.67.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 453). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.67.2 Constructor & Destructor Documentation

6.67.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::ActiveMQ () [inline]

6.67.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::~ActiveM () [inline, virtual]

6.67.3 Member Function Documentation

6.67.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.67.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.67.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.67.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867).

6.67.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868).

6.67.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

6.67.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTextMessageMarshaller.h**

6.68 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 457).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTextMessageMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual ~**ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.68.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 457). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.68.2 Constructor & Destructor Documentation

6.68.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::ActiveMQ () [inline]

6.68.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::~ActiveM () [inline, virtual]

6.68.3 Member Function Documentation

6.68.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.68.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.68.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.68.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872).

6.68.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873).

6.68.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

6.68.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTextMessageMarshaller.h**

6.69 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessag Class Reference

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 461).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller:

Public Member Functions

- **ActiveMQTextMessageMarshaller** ()
- virtual ~**ActiveMQTextMessageMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.69.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 461). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.69.2 Constructor & Destructor Documentation

6.69.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::ActiveMQ () [inline]

6.69.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::~ActiveM () [inline, virtual]

6.69.3 Member Function Documentation

6.69.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.69.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::getDataStru () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.69.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.69.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::looseUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862).

6.69.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863).

6.69.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightMarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

6.69.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller::tightUnmars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTextMessageMarshaller.h**

6.70 activemq::commands::ActiveMQTopic Class Reference

`#include <src/main/activemq/commands/ActiveMQTopic.h>`Inheritance diagram for activemq::commands::ActiveMQTopic:

Public Member Functions

- **ActiveMQTopic** ()
- **ActiveMQTopic** (const std::string &name)
- virtual ~**ActiveMQTopic** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **ActiveMQTopic** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **cms::Destination** * **getCMSDestination** () const
- virtual **cms::Destination::DestinationType** **getDestinationType** () const

  Retrieve the Destination Type for this Destination.

- virtual **cms::Destination** * **clone** () const

  Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)

  Copies the contents of the given Destination object to this one.

- virtual const **cms::CMSProperties** & **getCMSProperties** () const

  Retrieve any properties that might be part of the destination that was specified.

- virtual std::string **getTopicName** () const throw (cms::CMSException)

  Gets the name of this topic.

Static Public Attributes

- static const unsigned char **ID_ACTIVEMQTOPIC** = 101

6.70.1 Constructor & Destructor Documentation

6.70.1.1 activemq::commands::ActiveMQTopic::ActiveMQTopic ()

6.70.1.2 activemq::commands::ActiveMQTopic::ActiveMQTopic (const std::string & *name*)

6.70.1.3 virtual activemq::commands::ActiveMQTopic::~ActiveMQTopic () [virtual]

6.70.2 Member Function Documentation

6.70.2.1 virtual cms::Destination* activemq::commands::ActiveMQTopic::clone () const [inline, virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implements **cms::Destination** (p. 1191).

6.70.2.2 virtual ActiveMQTopic* activemq::commands::ActiveMQTopic::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 231).

6.70.2.3 virtual void activemq::commands::ActiveMQTopic::copy (const cms::Destination & *source*) [inline, virtual]

Copies the contents of the given Destination object to this one.

Parameters:

source The source Destination object.

6.70.2.4 virtual void activemq::commands::ActiveMQTopic::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 232).

6.70.2.5 virtual bool activemq::commands::ActiveMQTopic::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

References activemq::commands::ActiveMQDestination::equals().

6.70.2.6 virtual const cms::Destination* activemq::commands::ActiveMQTopic::getCMSDestination () const [inline, virtual]

Returns:

the **cms::Destination** (p. 1190) interface pointer that the objects that derive from this class implement.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.70.2.7 virtual const cms::CMSProperties& activemq::commands::ActiveMQTopic::getCMSProperties () const [inline, virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

const reference to a properties object.

Implements **cms::Destination** (p. 1191).

6.70.2.8 virtual unsigned char activemq::commands::ActiveMQTopic::getDataStructureType () const [virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 233).

6.70.2.9 virtual cms::Destination::DestinationType activemq::commands::ActiveMQTopic::getDestinationType () const [inline, virtual]

Retrieve the Destination Type for this Destination.

Returns:

The Destination Type

Implements **activemq::commands::ActiveMQDestination** (p. 234).

References cms::Destination::TOPIC.

6.70.2.10 virtual std::string activemq::commands::ActiveMQTopic::getTopicName () const throw (cms::CMSException) [inline, virtual]

Gets the name of this topic.

Returns:

The topic name.

Implements **cms::Topic** (p. 2571).

6.70.2.11 virtual std::string activemq::commands::ActiveMQTopic::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::ActiveMQDestination** (p. 237).

6.70.3 Field Documentation

6.70.3.1 const unsigned char activemq::commands::ActiveMQTopic::ID_ACTIVEMQTOPIC = 101 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ActiveMQTopic.h**

6.71 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarsh Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 469).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQTopicMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual ~**ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.71.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 469). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.71.2 Constructor & Destructor Documentation

6.71.2.1 activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::ActiveMQTopicM () [inline]

6.71.2.2 virtual activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::~ActiveMQTopicl () [inline, virtual]

6.71.3 Member Function Documentation

6.71.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.71.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.71.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

6.71.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241).

6.71.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

6.71.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 242).

6.71.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 243).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ActiveMQTopicMarshaller.h**

6.72 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarsh Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 473).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTopicMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual ~**ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.72.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 473). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.72.2 Constructor & Destructor Documentation

6.72.2.1 activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::ActiveMQTopicM () [inline]

6.72.2.2 virtual activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::~ActiveMQTopicl () [inline, virtual]

6.72.3 Member Function Documentation

6.72.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.72.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.72.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

6.72.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245).

6.72.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

6.72.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246).

6.72.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 247).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ActiveMQTopicMarshaller.h**

6.73 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarsh Class Reference

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 477).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTopicMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller:

Public Member Functions

- **ActiveMQTopicMarshaller** ()
- virtual ~**ActiveMQTopicMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.73.1 Detailed Description

Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 477). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.73.2 Constructor & Destructor Documentation

6.73.2.1 activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::ActiveMQTopicM () [inline]

6.73.2.2 virtual activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::~ActiveMQTopicl () [inline, virtual]

6.73.3 Member Function Documentation

6.73.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.73.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.73.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

6.73.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249).

6.73.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

6.73.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 250).

6.73.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 251).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ActiveMQTopicMarshaller.h**

6.74 activemq::core::ActiveMQTransactionContext Class Reference

Transaction Management class, hold messages that are to be redelivered upon a request to rollback.

`#include <src/main/activemq/core/ActiveMQTransactionContext.h>`

Public Member Functions

- **ActiveMQTransactionContext** (**ActiveMQSession** *session, const **decaf::util::Properties** &properties)

  Constructor.

- virtual ~**ActiveMQTransactionContext** ()
- virtual void **addSynchronization** (const **Pointer< Synchronization >** &sync)

  *Adds a **Synchronization** (p. 2516) to this Transaction.*

- virtual void **removeSynchronization** (const **Pointer< Synchronization >** &sync)

  *Removes a **Synchronization** (p. 2516) to this Transaction.*

- virtual void **begin** () throw (exceptions::ActiveMQException)

  Begins a new transaction if one is not currently in progress.

- virtual void **commit** () throw (exceptions::ActiveMQException)

  Commit the current Transaction.

- virtual void **rollback** () throw (exceptions::ActiveMQException)

  Rollback the current Transaction.

- virtual const **decaf::lang::Pointer< commands::TransactionId >** & **getTransactionId** () const

  Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

- virtual bool **isInTransaction** () const

  Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

- int **getMaximumRedeliveries** () const
- long long **getRedeliveryDelay** () const

6.74.1 Detailed Description

Transaction Management class, hold messages that are to be redelivered upon a request to rollback. The Transaction represents an always running transaction, when it is committed or rolled back it silently creates a new transaction for the next set of messages. The only way to permanently end this transaction is to delete it.

Configuration options

transaction.maxRedeliveryCount Max number of times a message can be re-delivered, if the session is rolled back more than this many time, the message is dropped.

6.74.2 Constructor & Destructor Documentation

6.74.2.1 activemq::core::ActiveMQTransactionContext::ActiveMQTransactionContext (ActiveMQSession * *session*, const decaf::util::Properties & *properties*)

Constructor.

Parameters:

session - the session that contains this transaction

properties - configuration parameters for this object

6.74.2.2 virtual activemq::core::ActiveMQTransactionContext::~ActiveMQTransactionContext () [virtual]

6.74.3 Member Function Documentation

6.74.3.1 virtual void activemq::core::ActiveMQTransactionContext::addSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Adds a **Synchronization** (p. 2516) to this Transaction.

Parameters:

sync - The **Synchronization** (p. 2516) instance to add.

6.74.3.2 virtual void activemq::core::ActiveMQTransactionContext::begin () throw (exceptions::ActiveMQException) [virtual]

Begins a new transaction if one is not currently in progress.

Exceptions:

ActiveMQException

6.74.3.3 virtual void activemq::core::ActiveMQTransactionContext::commit () throw (exceptions::ActiveMQException) [virtual]

Commit the current Transaction.

Exceptions:

ActiveMQException

6.74.3.4 int activemq::core::ActiveMQTransactionContext::getMaximumRedeliveries () const [inline]

6.74.3.5 long long activemq::core::ActiveMQTransactionContext::getRedeliveryDelay () const [inline]

6.74.3.6 virtual const decaf::lang::Pointer<commands::TransactionId>& activemq::core::ActiveMQTransactionContext::getTransactionId () const [inline, virtual]

Get the Transaction Id object for the current Transaction, returns NULL if no transaction is running.

Returns:

TransactionInfo

Exceptions:

InvalidStateException if a Transaction is not in progress.

6.74.3.7 virtual bool activemq::core::ActiveMQTransactionContext::isInTransaction () const [inline, virtual]

Checks to see if there is currently a Transaction in progress returns false if not, true otherwise.

Returns:

true if a transaction is in progress.

6.74.3.8 virtual void activemq::core::ActiveMQTransactionContext::removeSynchronization (const Pointer< Synchronization > & *sync*) [virtual]

Removes a **Synchronization** (p. 2516) to this Transaction.

Parameters:

sync - The **Synchronization** (p. 2516) instance to add.

6.74.3.9 virtual void activemq::core::ActiveMQTransactionContext::rollback () throw (exceptions::ActiveMQException) [virtual]

Rollback the current Transaction.

Exceptions:

ActiveMQException

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQTransactionContext.h**

6.75 decaf::lang::Appendable Class Reference

`#include <src/main/decaf/lang/Appendable.h>`Inheritance diagram for decaf::lang::Appendable:

Public Member Functions

- virtual ~**Appendable** ()
- virtual **Appendable** & **append** (char value)=0 throw (decaf::lang::Exception)

  *Appends the specified character to this **Appendable** (p. 484).*

- virtual **Appendable** & **append** (const **CharSequence** *csq)=0 throw (decaf::lang::Exception)

  *Appends the specified character sequence to this **Appendable** (p. 484).*

- virtual **Appendable** & **append** (const **CharSequence** *csq, std::size_t start, std::size_t end)=0 throw (decaf::lang::Exception)

  *Appends a subsequence of the specified character sequence to this **Appendable** (p. 484).*

6.75.1 Constructor & Destructor Documentation

6.75.1.1 virtual decaf::lang::Appendable::~Appendable () `[inline, virtual]`

6.75.2 Member Function Documentation

6.75.2.1 virtual Appendable& decaf::lang::Appendable::append (const CharSequence * *csq*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::Exception) `[pure virtual]`

Appends a subsequence of the specified character sequence to this **Appendable** (p. 484).

Parameters:

csq - The character sequence from which a subsequence will be appended. If csq is NULL, then characters will be appended as if csq contained the string "null".

start - The index of the first character in the subsequence

end - The index of the character following the last character in the subsequence

Returns:

a Reference to this **Appendable** (p. 484)

Exceptions:

Exception *(p. 1268)* if an error occurs.

IndexOutOfBoundsException start is greater than end, or end is greater than csq.length()

Implemented in **decaf::nio::CharBuffer** (p. 820).

6.75.2.2 virtual Appendable& decaf::lang::Appendable::append (const CharSequence * *csq*) throw (decaf::lang::Exception) [pure virtual]

Appends the specified character sequence to this **Appendable** (p. 484).

Parameters:

csq - The character sequence from which a subsequence will be appended. If csq is NULL, then characters will be appended as if csq contained the string "null".

Returns:

a Reference to this **Appendable** (p. 484)

Exceptions:

Exception (p. ***1268***) if an error occurs.

Implemented in **decaf::nio::CharBuffer** (p. 821).

6.75.2.3 virtual Appendable& decaf::lang::Appendable::append (char *value*) throw (decaf::lang::Exception) [pure virtual]

Appends the specified character to this **Appendable** (p. 484).

Parameters:

value - The character to append

Returns:

a Reference to this **Appendable** (p. 484)

Exceptions:

Exception (p. ***1268***) if an error occurs.

Implemented in **decaf::nio::CharBuffer** (p. 821).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Appendable.h**

6.76 decaf::internal::AprPool Class Reference

Wraps an APR pool object so that classes in **decaf** (p. 94) can create a static member for use in static methods where apr function calls that need a pool are made.

`#include <src/main/decaf/internal/AprPool.h>`

Public Member Functions

- **AprPool** ()
- virtual ~**AprPool** ()
- apr_pool_t * **getAprPool** () const

  *Gets the **internal** (p. 95) APR Pool.*

- void **cleanup** ()

  Clears data that was allocated by this pool.

Static Public Member Functions

- static apr_pool_t * **getGlobalPool** ()

  Gets a pointer to the Global APR Pool used for the Application.

6.76.1 Detailed Description

Wraps an APR pool object so that classes in **decaf** (p. 94) can create a static member for use in static methods where apr function calls that need a pool are made.

6.76.2 Constructor & Destructor Documentation

6.76.2.1 decaf::internal::AprPool::AprPool ()

6.76.2.2 virtual decaf::internal::AprPool::~AprPool () `[virtual]`

6.76.3 Member Function Documentation

6.76.3.1 void decaf::internal::AprPool::cleanup ()

Clears data that was allocated by this pool. Users should call this after getting the data from the APR functions and copying it to someplace safe.

6.76.3.2 apr_pool_t* decaf::internal::AprPool::getAprPool () const

Gets the **internal** (p. 95) APR Pool.

Returns:

the **internal** (p. 95) APR pool

6.76.3.3 static apr_pool_t* decaf::internal::AprPool::getGlobalPool () [static]

Gets a pointer to the Global APR Pool used for the Application.

Returns:

pointer to the global APR Pool

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**AprPool.h**

6.77 decaf::util::concurrent::atomic::AtomicBoolean Class Reference

A boolean value that may be updated atomically.

`#include <src/main/decaf/util/concurrent/atomic/AtomicBoolean.h>`

Public Member Functions

- **AtomicBoolean** ()
  *Creates a new **AtomicBoolean** (p. 488) whose initial value is false.*
- **AtomicBoolean** (bool initialValue)
  *Creates a new **AtomicBoolean** (p. 488) with the initial value.*
- virtual ~**AtomicBoolean** ()
- bool **get** () const
  *Gets the current value of this **AtomicBoolean** (p. 488).*
- void **set** (bool newValue)
  Unconditionally sets to the given value.
- bool **compareAndSet** (bool expect, bool update)
  Atomically sets the value to the given updated value if the current value == the expected value.
- bool **getAndSet** (bool newValue)
  Atomically sets to the given value and returns the previous value.
- std::string **toString** () const
  Returns the String representation of the current value.

6.77.1 Detailed Description

A boolean value that may be updated atomically. An **AtomicBoolean** (p. 488) is used in applications such as atomically updated flags, and cannot be used as a replacement for a Boolean.

6.77.2 Constructor & Destructor Documentation

6.77.2.1 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean ()

Creates a new **AtomicBoolean** (p. 488) whose initial value is false.

6.77.2.2 decaf::util::concurrent::atomic::AtomicBoolean::AtomicBoolean (bool *initialValue*)

Creates a new **AtomicBoolean** (p. 488) with the initial value.

Parameters:

initialValue - The initial value of this boolean.

6.77.2.3 virtual decaf::util::concurrent::atomic::AtomicBoolean::~AtomicBoolean () [inline, virtual]

6.77.3 Member Function Documentation

6.77.3.1 bool decaf::util::concurrent::atomic::AtomicBoolean::compareAndSet (bool *expect*, bool *update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.77.3.2 bool decaf::util::concurrent::atomic::AtomicBoolean::get () const [inline]

Gets the current value of this **AtomicBoolean** (p. 488).

Returns:

the currently set value.

6.77.3.3 bool decaf::util::concurrent::atomic::AtomicBoolean::getAndSet (bool *newValue*)

Atomically sets to the given value and returns the previous value.

Parameters:

newValue - the new value

Returns:

the previous value

6.77.3.4 void decaf::util::concurrent::atomic::AtomicBoolean::set (bool *newValue*) [inline]

Unconditionally sets to the given value.

Parameters:

newValue - the new value

6.77.3.5 std::string decaf::util::concurrent::atomic::AtomicBoolean::toString () const

Returns the String representation of the current value.

Returns:

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicBoolean.h**

6.78 decaf::util::concurrent::atomic::AtomicInteger Class Reference

An int value that may be updated atomically.

`#include <src/main/decaf/util/concurrent/atomic/AtomicInteger.h>`Inheritance diagram for decaf::util::concurrent::atomic::AtomicInteger:

Public Member Functions

- **AtomicInteger** ()

  *Create a new **AtomicInteger** (p. 491) with an initial value of 0.*

- **AtomicInteger** (int initialValue)

  *Create a new **AtomicInteger** (p. 491) with the given initial value.*

- virtual ~**AtomicInteger** ()
- int **get** () const

  Gets the current value.

- void **set** (int newValue)

  Sets to the given value.

- int **getAndSet** (int newValue)

  Atomically sets to the given value and returns the old value.

- bool **compareAndSet** (int expect, int update)

  Atomically sets the value to the given updated value if the current value == the expected value.

- int **getAndIncrement** ()

  Atomically increments by one the current value.

- int **getAndDecrement** ()

  Atomically decrements by one the current value.

- int **getAndAdd** (int delta)

  Atomically adds the given value to the current value.

- int **incrementAndGet** ()

  Atomically increments by one the current value.

- int **decrementAndGet** ()

  Atomically decrements by one the current value.

- int **addAndGet** (int delta)

  Atomically adds the given value to the current value.

- std::string **toString** () const

Returns the String representation of the current value.

- int **intValue** () const

  Description copied from class: Number Returns the value of the specified number as an int.

- long long **longValue** () const

  Description copied from class: Number Returns the value of the specified number as a long.

- float **floatValue** () const

  Description copied from class: Number Returns the value of the specified number as a float.

- double **doubleValue** () const

  Description copied from class: Number Returns the value of the specified number as a double.

6.78.1 Detailed Description

An int value that may be updated atomically. An **AtomicInteger** (p. 491) is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes.

6.78.2 Constructor & Destructor Documentation

6.78.2.1 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger ()

Create a new **AtomicInteger** (p. 491) with an initial value of 0.

6.78.2.2 decaf::util::concurrent::atomic::AtomicInteger::AtomicInteger (int *initialValue*)

Create a new **AtomicInteger** (p. 491) with the given initial value.

Parameters:

initialValue - The initial value of this object.

6.78.2.3 virtual decaf::util::concurrent::atomic::AtomicInteger::~AtomicInteger () [inline, virtual]

6.78.3 Member Function Documentation

6.78.3.1 int decaf::util::concurrent::atomic::AtomicInteger::addAndGet (int *delta*)

Atomically adds the given value to the current value.

Parameters:

delta - the value to add.

Returns:

the updated value.

6.78.3.2 bool decaf::util::concurrent::atomic::AtomicInteger::compareAndSet (int *expect*, int *update*)

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.78.3.3 int decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet ()

Atomically decrements by one the current value.

Returns:

the updated value.

Referenced by decaf::lang::AtomicRefCounter::release().

6.78.3.4 double decaf::util::concurrent::atomic::AtomicInteger::doubleValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a double. This may involve rounding.

Returns:

the numeric value represented by this object after conversion to type double.

Implements **decaf::lang::Number** (p. 1955).

6.78.3.5 float decaf::util::concurrent::atomic::AtomicInteger::floatValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a float. This may involve rounding.

Returns:

the numeric value represented by this object after conversion to type float.

Implements **decaf::lang::Number** (p. 1955).

6.78.3.6 int decaf::util::concurrent::atomic::AtomicInteger::get () const [inline]

Gets the current value.

Returns:

the current value.

6.78.3.7 int decaf::util::concurrent::atomic::AtomicInteger::getAndAdd (int *delta*)

Atomically adds the given value to the current value.

Parameters:

delta - The value to add.

Returns:

the previous value.

6.78.3.8 int decaf::util::concurrent::atomic::AtomicInteger::getAndDecrement ()

Atomically decrements by one the current value.

Returns:

the previous value.

6.78.3.9 int decaf::util::concurrent::atomic::AtomicInteger::getAndIncrement ()

Atomically increments by one the current value.

Returns:

the previous value.

6.78.3.10 int decaf::util::concurrent::atomic::AtomicInteger::getAndSet (int *newValue*)

Atomically sets to the given value and returns the old value.

Parameters:

newValue - the new value.

Returns:

the previous value.

6.78.3.11 int decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet ()

Atomically increments by one the current value.

Returns:

the updated value.

Referenced by decaf::lang::AtomicRefCounter::AtomicRefCounter().

6.78.3.12 int decaf::util::concurrent::atomic::AtomicInteger::intValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as an int. This may involve rounding or truncation.

Returns:

the numeric value represented by this object after conversion to type int.

Implements **decaf::lang::Number** (p. 1955).

6.78.3.13 long long decaf::util::concurrent::atomic::AtomicInteger::longValue () const [virtual]

Description copied from class: Number Returns the value of the specified number as a long. This may involve rounding or truncation.

Returns:

the numeric value represented by this object after conversion to type long long.

Implements **decaf::lang::Number** (p. 1955).

6.78.3.14 void decaf::util::concurrent::atomic::AtomicInteger::set (int *newValue*) [inline]

Sets to the given value.

Parameters:

newValue - the new value

6.78.3.15 std::string decaf::util::concurrent::atomic::AtomicInteger::toString () const

Returns the String representation of the current value.

Returns:

the String representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicInteger.h**

6.79 decaf::lang::AtomicRefCounter Class Reference

`#include <src/main/decaf/lang/Pointer.h>`Inheritance diagram for decaf::lang::AtomicRefCounter:

Public Member Functions

- **AtomicRefCounter** ()
- **AtomicRefCounter** (const **AtomicRefCounter** &other)
- void **swap** (**AtomicRefCounter** &other)

  Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

- bool **release** ()

  *Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the **internal** (p. 95) counter is destroyed and this instance is now considered to be unreferenced.*

6.79.1 Constructor & Destructor Documentation

6.79.1.1 decaf::lang::AtomicRefCounter::AtomicRefCounter () [inline]

6.79.1.2 decaf::lang::AtomicRefCounter::AtomicRefCounter (const AtomicRefCounter & *other*) [inline]

References decaf::util::concurrent::atomic::AtomicInteger::incrementAndGet().

6.79.2 Member Function Documentation

6.79.2.1 bool decaf::lang::AtomicRefCounter::release () [inline]

Removes a reference to the counter Atomically and returns if the counter has reached zero, once the counter hits zero, the **internal** (p. 95) counter is destroyed and this instance is now considered to be unreferenced.

Returns:

true if the count is now zero.

References decaf::util::concurrent::atomic::AtomicInteger::decrementAndGet().

6.79.2.2 void decaf::lang::AtomicRefCounter::swap (AtomicRefCounter & *other*) [inline]

Swaps this instance's reference counter with the one given, this allows for copy-and-swap semantics of this object.

Parameters:

other The value to swap with this one's.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.80 decaf::util::concurrent::atomic::AtomicReference< T > Class Template Reference

An Pointer reference that may be updated atomically.

`#include <src/main/decaf/util/concurrent/atomic/AtomicReference.h>`

Public Member Functions

- **AtomicReference** ()
- **AtomicReference** (T *value)
- virtual ~**AtomicReference** ()
- T * **get** () const

  Gets the Current Value.

- void **set** (T *newValue)

  Sets the Current value of this Reference.

- bool **compareAndSet** (T *expect, T *update)

  Atomically sets the value to the given updated value if the current value == the expected value.

- T * **getAndSet** (T *newValue)

  Atomically sets to the given value and returns the old value.

- std::string **toString** () const

  Returns the String representation of the current value.

6.80.1 Detailed Description

template<typename T> class decaf::util::concurrent::atomic::AtomicReference< T >

An Pointer reference that may be updated atomically.

6.80.2 Constructor & Destructor Documentation

6.80.2.1 template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference () [inline]

6.80.2.2 template<typename T > decaf::util::concurrent::atomic::AtomicReference< T >::AtomicReference (T * *value*) [inline]

6.80.2.3 template<typename T > virtual decaf::util::concurrent::atomic::AtomicReference< T >::~AtomicReference () [inline, virtual]

6.80.3 Member Function Documentation

6.80.3.1 template<typename T > bool decaf::util::concurrent::atomic::AtomicReference< T >::compareAndSet (T * *expect*, T * *update*) [inline]

Atomically sets the value to the given updated value if the current value == the expected value.

Parameters:

expect - the expected value

update - the new value

Returns:

true if successful. False return indicates that the actual value was not equal to the expected value.

6.80.3.2 template<typename T > T* decaf::util::concurrent::atomic::AtomicReference< T >::get () const [inline]

Gets the Current Value.

Returns:

the current value of this Reference.

6.80.3.3 template<typename T > T* decaf::util::concurrent::atomic::AtomicReference< T >::getAndSet (T * *newValue*) [inline]

Atomically sets to the given value and returns the old value.

Parameters:

newValue- the new value

Returns:

the previous value.

6.80.3.4 template<typename T > void decaf::util::concurrent::atomic::AtomicReference< T >::set (T * *newValue*) [inline]

Sets the Current value of this Reference.

Parameters:

newValue The new Value of this Reference.

6.80.3.5 template<typename T > std::string decaf::util::concurrent::atomic::AtomicReference< T >::toString () const [inline]

Returns the String representation of the current value.

Returns:

string representation of the current value.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/atomic/**AtomicReference.h**

6.81 activemq::transport::failover::BackupTransport Class Reference

`#include <src/main/activemq/transport/failover/BackupTransport.h>`Inheritance diagram for activemq::transport::failover::BackupTransport:

Public Member Functions

- **BackupTransport** (**BackupTransportPool** *failover)
- virtual ~**BackupTransport** ()
- **decaf::net::URI getUri** () const

  Gets the URI assigned to this Backup.

- void **setUri** (const **decaf::net::URI** &uri)

  *Sets the URI assigned to this **Transport** (p. 2608).*

- const **Pointer**< **Transport** > & **getTransport** ()

  *Gets the currently held **transport** (p. 67).*

- void **setTransport** (const **Pointer**< **Transport** > &transport)

  *Sets the held **transport** (p. 67), if not NULL then start to listen for **exceptions** (p. 62) from the held **transport** (p. 67).*

- virtual void **onException** (const **decaf::lang::Exception** &ex)

  *Event handler for an exception from a command **transport** (p. 67).*

- bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- void **setClosed** (bool value)

  *Sets the closed flag on this **Transport** (p. 2608).*

6.81.1 Constructor & Destructor Documentation

6.81.1.1 activemq::transport::failover::BackupTransport::BackupTransport (BackupTransportPool * *failover*)

6.81.1.2 virtual activemq::transport::failover::BackupTransport::~BackupTransport () `[virtual]`

6.81.2 Member Function Documentation

6.81.2.1 const Pointer<Transport>& activemq::transport::failover::BackupTransport::getTransport () `[inline]`

Gets the currently held **transport** (p. 67).

Returns:

pointer to the held **transport** (p. 67) or NULL if not set.

6.81.2.2 decaf::net::URI activemq::transport::failover::BackupTransport::getUri () const [inline]

Gets the URI assigned to this Backup.

Returns:

the assigned URI

6.81.2.3 bool activemq::transport::failover::BackupTransport::isClosed () const [inline]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

6.81.2.4 virtual void activemq::transport::failover::BackupTransport::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67). The **BackupTransport** (p. 501) closes its internal **Transport** (p. 2608) when an exception is received and returns the URI to the pool of URIs to attempt connections to.

Parameters:

ex The exception that was passed to this listener to handle.

Implements **activemq::transport::TransportListener** (p. 2625).

6.81.2.5 void activemq::transport::failover::BackupTransport::setClosed (bool *value*) [inline]

Sets the closed flag on this **Transport** (p. 2608).

Parameters:

value - true for closed.

6.81.2.6 void activemq::transport::failover::BackupTransport::setTransport (const Pointer< Transport > & *transport*) [inline]

Sets the held **transport** (p. 67), if not NULL then start to listen for **exceptions** (p. 62) from the held **transport** (p. 67).

Parameters:

transport *(p. 67)* The **transport** (p. 67) to hold.

6.81.2.7 void activemq::transport::failover::BackupTransport::setUri (const decaf::net::URI & *uri*) [inline]

Sets the URI assigned to this **Transport** (p. 2608).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransport.h**

6.82 activemq::transport::failover::BackupTransportPool Class Reference

`#include <src/main/activemq/transport/failover/BackupTransportPool.h>`Inheritance diagram for activemq::transport::failover::BackupTransportPool:

Public Member Functions

- **BackupTransportPool** (const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- **BackupTransportPool** (int backupPoolSize, const **Pointer**< **CompositeTaskRunner** > &taskRunner, const **Pointer**< **CloseTransportsTask** > &closeTask, const **Pointer**< **URIPool** > &uriPool)
- virtual ~**BackupTransportPool** ()
- virtual bool **isPending** () const

  Return true if we don't currently have enough Connected Transports.

- **Pointer**< **BackupTransport** > **getBackup** ()

  *Get a Connected **Transport** (p. 2608) from the pool of Backups if any are present, otherwise it return a NULL Pointer.*

- virtual bool **iterate** ()

  Connect to a Backup Broker if we haven't already connected to the max number of Backups.

- int **getBackupPoolSize** () const

  Gets the Max number of Backups this Task will create.

- void **setBackupPoolSize** (int size)

  Sets the Max number of Backups this Task will create.

- bool **isEnabled** () const

  *Gets if the backup **Transport** (p. 2608) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.*

- void **setEnabled** (bool value)

  *Sets if this Backup **Transport** (p. 2608) Pool is enabled.*

Friends

- class **BackupTransport**

6.82.1 Constructor & Destructor Documentation

6.82.1.1 activemq::transport::failover::BackupTransportPool::BackupTransportPool (const Pointer< CompositeTaskRunner > & *taskRunner*, const Pointer< CloseTransportsTask > & *closeTask*, const Pointer< URIPool > & *uriPool*)

6.82.1.2 activemq::transport::failover::BackupTransportPool::BackupTransportPool (int *backupPoolSize*, const Pointer< CompositeTaskRunner > & *taskRunner*, const Pointer< CloseTransportsTask > & *closeTask*, const Pointer< URIPool > & *uriPool*)

6.82.1.3 virtual activemq::transport::failover::BackupTransportPool::~BackupTransportPool () [virtual]

6.82.2 Member Function Documentation

6.82.2.1 Pointer<BackupTransport> activemq::transport::failover::BackupTransportPool::getBackup ()

Get a Connected **Transport** (p. 2608) from the pool of Backups if any are present, otherwise it return a NULL Pointer.

Returns:

Pointer to a Connected **Transport** (p. 2608) or NULL

6.82.2.2 int activemq::transport::failover::BackupTransportPool::getBackupPoolSize () const [inline]

Gets the Max number of Backups this Task will create.

Returns:

the max number of active BackupTransports that will be created.

6.82.2.3 bool activemq::transport::failover::BackupTransportPool::isEnabled () const [inline]

Gets if the backup **Transport** (p. 2608) Pool has been enabled or not, when not enabled no backups are created and any that were are destroyed.

Returns:

true if enable.

6.82.2.4 virtual bool activemq::transport::failover::BackupTransportPool::isPending () const [virtual]

Return true if we don't currently have enough Connected Transports.

Implements **activemq::threads::CompositeTask** (p. 906).

6.82.2.5 virtual bool activemq::transport::failover::BackupTransportPool::iterate () [virtual]

Connect to a Backup Broker if we haven't already connected to the max number of Backups.

Implements **activemq::threads::Task** (p. 2520).

6.82.2.6 void activemq::transport::failover::BackupTransportPool::setBackupPoolSize (int *size*) [inline]

Sets the Max number of Backups this Task will create.

Parameters:

size - the max number of active BackupTransports that will be created.

6.82.2.7 void activemq::transport::failover::BackupTransportPool::setEnabled (bool *value*)

Sets if this Backup **Transport** (p. 2608) Pool is enabled. When not enabled no Backups are created and any that were are destroyed.

Parameters:

value - true to enable backup creation, false to disable.

6.82.3 Friends And Related Function Documentation

6.82.3.1 friend class BackupTransport [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**BackupTransportPool.h**

6.83 activemq::commands::BaseCommand Class Reference

`#include <src/main/activemq/commands/BaseCommand.h>`Inheritance diagram for activemq::commands::BaseCommand:

Public Member Functions

- **BaseCommand** ()
- virtual ~**BaseCommand** ()
- virtual void **setCommandId** (int id)

  *Sets the **Command** (p. 879) Id of this **Message** (p. 1737).*

- virtual int **getCommandId** () const

  *Gets the **Command** (p. 879) Id of this **Message** (p. 1737).*

- virtual void **setResponseRequired** (const bool required)

  *Set if this **Message** (p. 1737) requires a **Response** (p. 2231).*

- virtual bool **isResponseRequired** () const

  *Is a **Response** (p. 2231) required for this **Command** (p. 879).*

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual bool **isConnectionInfo** () const
- virtual bool **isConsumerInfo** () const
- virtual bool **isBrokerInfo** () const
- virtual bool **isMessage** () const
- virtual bool **isMessageAck** () const
- virtual bool **isKeepAliveInfo** () const
- virtual bool **isMessageDispatch** () const
- virtual bool **isMessageDispatchNotification** () const
- virtual bool **isProducerAck** () const
- virtual bool **isProducerInfo** () const
- virtual bool **isResponse** () const
- virtual bool **isRemoveInfo** () const
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual bool **isShutdownInfo** () const
- virtual bool **isTransactionInfo** () const
- virtual bool **isWireFormatInfo** () const

6.83.1 Constructor & Destructor Documentation

6.83.1.1 activemq::commands::BaseCommand::BaseCommand () [inline]

6.83.1.2 virtual activemq::commands::BaseCommand::~BaseCommand () [inline, virtual]

6.83.2 Member Function Documentation

6.83.2.1 virtual void activemq::commands::BaseCommand::copyDataStructure (const DataStructure * *src*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 147), **activemq::commands::ActiveMQBytesMessage** (p. 167), **activemq::commands::ActiveMQMapMessage** (p. 258), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 377), **activemq::commands::ActiveMQTextMessage** (p. 450), **activemq::commands::BrokerError** (p. 587), **activemq::commands::BrokerInfo** (p. 609), **activemq::commands::ConnectionControl** (p. 946), **activemq::commands::ConnectionError** (p. 963), **activemq::commands::ConnectionInfo** (p. 999), **activemq::commands::ConsumerControl** (p. 1029), **activemq::commands::ConsumerInfo** (p. 1064), **activemq::commands::ControlCommand** (p. 1084), **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::DestinationInfo** (p. 1195), **activemq::commands::ExceptionResponse** (p. 1277), **activemq::commands::FlushCommand** (p. 1358), **activemq::commands::IntegerResponse** (p. 1443), **activemq::commands::KeepAliveInfo** (p. 1556), **activemq::commands::Message** (p. 1742), **activemq::commands::MessageAck** (p. 1778), **activemq::commands::MessageDispatch** (p. 1801), **activemq::commands::MessageDispatchNotification** (p. 1824), **activemq::commands::MessagePull** (p. 1894), **activemq::commands::ProducerAck** (p. 2087), **activemq::commands::ProducerInfo** (p. 2123), **activemq::commands::RemoveInfo** (p. 2178), **activemq::commands::RemoveSubscriptionInfo** (p. 2194), **activemq::commands::ReplayCommand** (p. 2211), **activemq::commands::Response** (p. 2232), **activemq::commands::SessionInfo** (p. 2301), **activemq::commands::ShutdownInfo** (p. 2350), **activemq::commands::TransactionInfo** (p. 2590), and **activemq::commands::WireFormatInfo** (p. 2702).

References getCommandId(), and isResponseRequired().

6.83.2.2 virtual bool activemq::commands::BaseCommand::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 148), **activemq::commands::ActiveMQBytesMessage** (p. 167), **activemq::commands::ActiveMQMapMessage** (p. 259), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 377), **activemq::commands::ActiveMQTextMessage** (p. 450), **activemq::commands::BrokerInfo** (p. 609), **activemq::commands::ConnectionControl** (p. 947), **activemq::commands::ConnectionError** (p. 963), **activemq::commands::ConnectionInfo** (p. 999), **activemq::commands::ConsumerControl** (p. 1030), **activemq::commands::ConsumerInfo** (p. 1064), **activemq::commands::ControlCommand** (p. 1084), **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::DestinationInfo** (p. 1196), **activemq::commands::ExceptionResponse** (p. 1277), **activemq::commands::FlushCommand** (p. 1358), **activemq::commands::IntegerResponse** (p. 1443), **activemq::commands::KeepAliveInfo** (p. 1556), **activemq::commands::Message** (p. 1742), **activemq::commands::MessageAck** (p. 1779), **activemq::commands::MessageDispatch** (p. 1802), **activemq::commands::MessageDispatchNotification** (p. 1825), **activemq::commands::MessagePull** (p. 1895), **activemq::commands::ProducerAck** (p. 2087), **activemq::commands::ProducerInfo** (p. 2124), **activemq::commands::RemoveInfo** (p. 2178), **activemq::commands::RemoveSubscriptionInfo** (p. 2195), **activemq::commands::ReplayCommand** (p. 2211), **activemq::commands::Response** (p. 2232), **activemq::commands::SessionInfo** (p. 2301), **activemq::commands::ShutdownInfo** (p. 2350), **activemq::commands::TransactionInfo** (p. 2590), and **activemq::commands::WireFormatInfo** (p. 2702).

References activemq::commands::BaseDataStructure::equals().

6.83.2.3 virtual int activemq::commands::BaseCommand::getCommandId () const [inline, virtual]

Gets the **Command** (p. 879) Id of this **Message** (p. 1737).

Returns:

Command (p. 879) Id

Implements **activemq::commands::Command** (p. 880).

Referenced by copyDataStructure().

6.83.2.4 virtual bool activemq::commands::BaseCommand::isBrokerInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 880).

Reimplemented in **activemq::commands::BrokerInfo** (p. 611).

6.83.2.5 virtual bool activemq::commands::BaseCommand::isConnectionInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 880).

Reimplemented in **activemq::commands::ConnectionInfo** (p. 1000).

6.83.2.6 virtual bool activemq::commands::BaseCommand::isConsumerInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 880).

Reimplemented in **activemq::commands::ConsumerInfo** (p. 1066).

6.83.2.7 virtual bool activemq::commands::BaseCommand::isKeepAliveInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 880).

Reimplemented in **activemq::commands::KeepAliveInfo** (p. 1557).

6.83.2.8 virtual bool activemq::commands::BaseCommand::isMessage () const [inline, virtual]

Implements **activemq::commands::Command** (p. 880).

Reimplemented in **activemq::commands::Message** (p. 1747).

6.83.2.9 virtual bool activemq::commands::BaseCommand::isMessageAck () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::MessageAck** (p. 1780).

6.83.2.10 virtual bool activemq::commands::BaseCommand::isMessageDispatch () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::MessageDispatch** (p. 1803).

6.83.2.11 virtual bool activemq::commands::BaseCommand::isMessageDispatchNotification () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::MessageDispatchNotification** (p. 1826).

6.83.2.12 virtual bool activemq::commands::BaseCommand::isProducerAck () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::ProducerAck** (p. 2088).

6.83.2.13 virtual bool activemq::commands::BaseCommand::isProducerInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::ProducerInfo** (p. 2125).

6.83.2.14 virtual bool activemq::commands::BaseCommand::isRemoveInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::RemoveInfo** (p. 2179).

6.83.2.15 virtual bool activemq::commands::BaseCommand::isRemoveSubscriptionInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::RemoveSubscriptionInfo** (p. 2196).

6.83.2.16 virtual bool activemq::commands::BaseCommand::isResponse () const [inline, virtual]

Implements **activemq::commands::Command** (p. 881).

Reimplemented in **activemq::commands::Response** (p. 2233).

6.83.2.17 virtual bool activemq::commands::BaseCommand::isResponseRequired () const [inline, virtual]

Is a **Response** (p. 2231) required for this **Command** (p. 879).

Returns:

true if a response is required.

Implements **activemq::commands::Command** (p. 882).

Referenced by copyDataStructure().

6.83.2.18 virtual bool activemq::commands::BaseCommand::isShutdownInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 882).

Reimplemented in **activemq::commands::ShutdownInfo** (p. 2351).

6.83.2.19 virtual bool activemq::commands::BaseCommand::isTransactionInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 882).

Reimplemented in **activemq::commands::TransactionInfo** (p. 2591).

6.83.2.20 virtual bool activemq::commands::BaseCommand::isWireFormatInfo () const [inline, virtual]

Implements **activemq::commands::Command** (p. 882).

Reimplemented in **activemq::commands::WireFormatInfo** (p. 2706).

6.83.2.21 virtual void activemq::commands::BaseCommand::setCommandId (int *id*) [inline, virtual]

Sets the **Command** (p. 879) Id of this **Message** (p. 1737).

Parameters:

id **Command** (p. 879) Id

Implements **activemq::commands::Command** (p. 882).

6.83.2.22 virtual void activemq::commands::BaseCommand::setResponseRequired (const bool *required*) [inline, virtual]

Set if this **Message** (p. 1737) requires a **Response** (p. 2231).

Parameters:

required true if response is required

Implements **activemq::commands::Command** (p. 882).

6.83.2.23 virtual std::string activemq::commands::BaseCommand::toString () const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implements **activemq::commands::Command** (p. 883).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 150), **activemq::commands::ActiveMQBytesMessage** (p. 173), **activemq::commands::ActiveMQMapMessage** (p. 266), **activemq::commands::ActiveMQMessage** (p. 281), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQStreamMessage** (p. 382), **activemq::commands::ActiveMQTextMessage** (p. 452), **activemq::commands::BrokerInfo**

(p. 612), **activemq::commands::ConnectionControl** (p. 948), **activemq::commands::ConnectionError** (p. 964), **activemq::commands::ConnectionInfo** (p. 1001), **activemq::commands::ConsumerControl** (p. 1031), **activemq::commands::ConsumerInfo** (p. 1067), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::DestinationInfo** (p. 1197), **activemq::commands::ExceptionResponse** (p. 1278), **activemq::commands::FlushCommand** (p. 1359), **activemq::commands::IntegerResponse** (p. 1444), **activemq::commands::KeepAliveInfo** (p. 1557), **activemq::commands::Message** (p. 1749), **activemq::commands::MessageAck** (p. 1781), **activemq::commands::MessageDispatch** (p. 1803), **activemq::commands::MessageDispatchNotification** (p. 1827), **activemq::commands::MessagePull** (p. 1896), **activemq::commands::ProducerAck** (p. 2088), **activemq::commands::ProducerInfo** (p. 2125), **activemq::commands::RemoveInfo** (p. 2179), **activemq::commands::RemoveSubscriptionInfo** (p. 2196), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2233), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2351), **activemq::commands::TransactionInfo** (p. 2592), and **activemq::commands::WireFormatInfo** (p. 2708).

References activemq::commands::BaseDataStructure::toString().

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BaseCommand.h**

6.84 activemq::wireformat::openwire::marshal::v3::BaseCommandMarsh Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 514).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.84.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 514). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.84.2 Constructor & Destructor Documentation

6.84.2.1 activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::BaseCommandMa () [inline]

6.84.2.2 virtual activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::~BaseCommandM () [inline, virtual]

6.84.3 Member Function Documentation

6.84.3.1 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 152), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 268), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 283), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 314), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 454), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 951), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 967), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1004), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1034), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1071), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1092), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1200), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1365), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1788), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

(p. 1838), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1903), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2091), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2132), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2190), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2203), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2219), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2313), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2361), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2599).

6.84.3.2 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 153), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 269), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 284), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 315), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 455), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 617), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 952), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 968), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1005), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1035), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1093), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1106), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1137), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1201), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1281), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1366), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1455), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1560),

activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 1789), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1817), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1839), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1904), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2092), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2133), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2191), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2204), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2220), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2314), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2362), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2600).

6.84.3.3 virtual int activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 153), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 269), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 284), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 315), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 389), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 455), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 617), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 952), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 968), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1005), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1035), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1072), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1093), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1106),

activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1137), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1201), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1281), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1366), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1455), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1560), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1789), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1817), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1839), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1868), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1904), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2092), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2133), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2191), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2204), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2220), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2314), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2362), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2600).

6.84.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 154), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 270), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 316), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 456), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 618), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 969), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1006),

activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1036), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1094), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1107), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1138), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1202), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1367), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1456), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1561), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1818), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1840), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1905), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2093), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2134), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2192), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2205), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2221), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2249), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2315), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2363), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2601).

6.84.3.5 virtual void activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 154), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 270), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 316), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 390), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

(p. 456), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 618), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 953), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 969), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1006), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1036), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1073), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1094), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1107), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1138), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1202), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1282), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1367), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1456), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1561), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1818), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1840), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1905), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2093), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2134), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2192), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2205), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2221), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2315), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2363), and **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2601).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BaseCommandMarshaller.h**

6.85 activemq::wireformat::openwire::marshal::v1::BaseCommandMarsh Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 521).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.85.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 521). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.85.2 Constructor & Destructor Documentation

6.85.2.1 activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::BaseCommandMa () [inline]

6.85.2.2 virtual activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::~BaseCommandM () [inline, virtual]

6.85.3 Member Function Documentation

6.85.3.1 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 156), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 272), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 458), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 620), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 955), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 971), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1008), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1038), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1088), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1204), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1361), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1792), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1820), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

(p. 1834), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1907), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2095), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 2136), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 2186), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2207), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2223), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 2305), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2357), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 2603).

6.85.3.2 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 459), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 621), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 956), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 972), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1009), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1039), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1076), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1089), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1110), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1141), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1205), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1285), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1362), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1451), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1568),

activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 1793), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1821), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1835), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1908), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2096), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 2137), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 2187), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2208), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2224), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 2306), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2358), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 2604).

6.85.3.3 virtual int activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 459), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 621), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 956), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 972), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1009), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1039), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1076), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1089), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1110),

activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1141), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1205), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1285), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1362), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1451), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1568), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1793), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1821), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1835), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1873), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1908), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2096), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 2137), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 2187), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2208), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2224), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 2306), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2358), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 2604).

6.85.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 957), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 973), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1010),

activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1040), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1077), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1090), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1206), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1363), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1569), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1794), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1822), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1836), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1909), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2097), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 2138), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 2188), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2209), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2225), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2254), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 2307), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2359), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 2605).

6.85.3.5 virtual void activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

(p. 460), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 957), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 973), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1010), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1040), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1077), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1090), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1206), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1363), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1569), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1794), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1822), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1836), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1874), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1909), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2097), **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller** (p. 2138), **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller** (p. 2188), **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller** (p. 2209), **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller** (p. 2225), **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2255), **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller** (p. 2307), **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller** (p. 2359), and **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller** (p. 2605).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BaseCommandMarshaller.h**

6.86 activemq::wireformat::openwire::marshal::v2::BaseCommandMarsh Class Reference

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 528).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller:

Public Member Functions

- **BaseCommandMarshaller** ()
- virtual ~**BaseCommandMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.86.1 Detailed Description

Marshaling code for Open Wire Format for **BaseCommandMarshaller** (p. 528). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.86.2 Constructor & Destructor Documentation

6.86.2.1 activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::BaseCommandMa () [inline]

6.86.2.2 virtual activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::~BaseCommandM () [inline, virtual]

6.86.3 Member Function Documentation

6.86.3.1 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 160), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 276), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 291), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 322), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 396), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 462), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 624), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 959), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 975), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1012), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1042), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1079), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1096), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1113), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1144), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1208), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1288), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1369), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1446), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1563), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1784), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1812), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

(p. 1830), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1899), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2099), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2128), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2182), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2199), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2215), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2309), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2353), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2595).

6.86.3.2 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 277), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 397), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 625), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 960), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 976), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1013), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1043), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1080), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1097), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1114), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1145), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1209), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1289), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1370), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1447), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1564),

activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 1785), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1813), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1831), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1900), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2100), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2129), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2183), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2200), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2216), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2310), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2354), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2596).

6.86.3.3 virtual int activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 277), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 397), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 625), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 960), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 976), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1013), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1043), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1080), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1097), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1114),

activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1145), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1209), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1289), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1370), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1447), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1785), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1813), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1831), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1863), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1900), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2100), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2129), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2183), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2200), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2216), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2310), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2354), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2596).

6.86.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 162), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 278), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 324), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 398), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 464), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 626), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 961), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 977), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1014),

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1044), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1081), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1098), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1115), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1210), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1290), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1371), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1448), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1565), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1786), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1814), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1832), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1901), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2101), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2130), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2184), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2201), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2217), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2244), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2311), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2355), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2597).

6.86.3.5 virtual void activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 162), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 278), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 324), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 398), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

(p. 464), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 626), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 961), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 977), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1014), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1044), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1081), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1098), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1115), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1210), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1290), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1371), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1448), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1565), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1786), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1814), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1832), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1864), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1901), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2101), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2130), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2184), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2201), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2217), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2245), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2311), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2355), and **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2597).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BaseCommandMarshaller.h**

6.87 activemq::wireformat::openwire::marshal::BaseDataStreamMarshal Class Reference

Base class for all Marshallers that **marshal** (p. 76) DataStructures to and from the wire using the OpenWire protocol.

`#include <src/main/activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>`

Inherits **activemq::wireformat::openwire::marshal::DataStreamMarshaller**.

Inherited by **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**, **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**, **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**, **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**, **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**, **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**, **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**, **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**, **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**, **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**, **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**, **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**, **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**, **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**, **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**, **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**, **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**, **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**, **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**, **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**, **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**, **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**, **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**, **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**, **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**, **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**, **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**, **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**, **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**, **ac-**

tivemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller, **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**, **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**, **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**, **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**, **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**, **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**, and **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**.

Public Member Functions

- virtual ~**BaseDataStreamMarshaller** ()
- virtual int **tightMarshal1** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (decaf::io::IOException)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (decaf::io::IOException)

  Tight Marshal to the given stream.

- virtual void **tightUnmarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED, **utils::BooleanStream** *bs AMQCPP_UNUSED) throw (decaf::io::IOException)

  Tight Un-Marshal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataOutputStream** *ds AMQCPP_UNUSED) throw (decaf::io::IOException)

  Tight Marshal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *format AMQCPP_UNUSED, **commands::DataStructure** *command AMQCPP_UNUSED, **decaf::io::DataInputStream** *dis AMQCPP_UNUSED) throw (decaf::io::IOException)

  Loose Un-Marshal to the given stream.

Static Public Member Functions

- static std::string **toString** (const **commands::MessageId** *id)

  Converts the object to a String.

- static std::string **toString** (const **commands::ProducerId** *id)

  Converts the object to a String.

- static std::string **toString** (const **commands::TransactionId** *txnId)

  Converts the given transaction ID into a String.

- static std::string **toHexFromBytes** (const std::vector< unsigned char > &data)

  given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Protected Member Functions

- virtual **commands::DataStructure** * **tightUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Unmarshal the cached object.

- virtual int **tightMarshalCachedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

- virtual void **tightMarshalCachedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

- virtual void **looseMarshalCachedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

- virtual **commands::DataStructure** * **looseUnmarshalCachedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Loose Unmarshal the cached object.

- virtual int **tightMarshalNestedObject1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

- virtual void **tightMarshalNestedObject2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

- virtual **commands::DataStructure** * **tightUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Unmarshal the nested object.

- virtual **commands::DataStructure** * **looseUnmarshalNestedObject** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Loose Unmarshal the nested object.

- virtual void **looseMarshalNestedObject** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *object, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Loose marshall the nested object.

- virtual std::string **tightUnmarshalString** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Performs Tight Unmarshaling of String Objects.

- virtual int **tightMarshalString1** (const std::string &value, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

- virtual void **tightMarshalString2** (const std::string &value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Marshals the passed string to the streams passed.

- virtual void **looseMarshalString** (const std::string value, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Loose Marshal the String to the DataOuputStream passed.

- virtual std::string **looseUnmarshalString** (**decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Loose Un-Marshal the String to the DataOuputStream passed.

- virtual int **tightMarshalLong1** (**OpenWireFormat** *wireFormat, long long value, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  *Tightly **marshal** (p. 76) the long long to the BooleanStream passed.*

- virtual void **tightMarshalLong2** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  *Tightly **marshal** (p. 76) the long long to the Streams passed.*

- virtual long long **tightUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  *Tight **marshal** (p. 76) the long long type.*

- virtual void **looseMarshalLong** (**OpenWireFormat** *wireFormat, long long value, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  *Tightly **marshal** (p. 76) the long long to the BooleanStream passed.*

- virtual long long **looseUnmarshalLong** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  *Loose **marshal** (p. 76) the long long type.*

- virtual std::vector< unsigned char > **tightUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Unmarshal an array of char.

- virtual std::vector< unsigned char > **looseUnmarshalByteArray** (**decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Loose Unmarshal an array of char.

- virtual std::vector< unsigned char > **tightUnmarshalConstByteArray** (**decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs, int size) throw (decaf::io::IOException)

  Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual std::vector< unsigned char > **looseUnmarshalConstByteArray** (**decaf::io::DataInputStream** *dataIn, int size) throw (decaf::io::IOException)

  Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

- virtual **commands::DataStructure** * **tightUnmarshalBrokerError** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Unarshall the Error object.

- virtual int **tightMarshalBrokerError1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Marshal the Error object.

- virtual void **tightMarshalBrokerError2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tight Marshal the Error object.

- virtual **commands::DataStructure** * **looseUnmarshalBrokerError** (**OpenWireFormat** *wireFormat, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Loose Unarshal the Error object.

- virtual void **looseMarshalBrokerError** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *data, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Tight Marshal the Error object.

- template<typename T >
  int **tightMarshalObjectArray1** (**OpenWireFormat** *wireFormat, std::vector< T > objects, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

- template<typename T >
  void **tightMarshalObjectArray2** (**OpenWireFormat** *wireFormat, std::vector< T > objects, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- template<typename T >
  void **looseMarshalObjectArray** (**OpenWireFormat** *wireFormat, std::vector< T > objects, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

- virtual std::string **readAsciiString** (**decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Given an DataInputStream read a know ASCII formatted string from the input and return that string.

6.87.1 Detailed Description

Base class for all Marshallers that **marshal** (p. 76) DataStructures to and from the wire using the OpenWire protocol.

Since:

2.0

6.87.2 Constructor & Destructor Documentation

6.87.2.1 virtual activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::~BaseDataStreamM () [inline, virtual]

6.87.3 Member Function Documentation

6.87.3.1 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

ds - DataOutputStream to **marshal** (p. 76) to

Exceptions:

IOException if an error occurs.

6.87.3.2 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalBrokerEr (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

Exceptions:

IOException if an error occurs.

6.87.3.3 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalCachedOl (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Loosely marshals the passed DataStructure based object to the passed stream returning nothing.

Parameters:

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to **marshal** (p. 76)

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.87.3.4 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalLong (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Tightly **marshal** (p. 76) the long long to the BooleanStream passed.

Parameters:

wireFormat - The OpenwireFormat properties

value - long long to **marshal** (p. 76)

dataOut - DataOutputStream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.5 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalNestedOb (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Loose marshall the nested object.

Parameters:

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to **marshal** (p. 76)

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.87.3.6 template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectAr (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [inline, protected]

Loosely Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters:

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

dataOut - stream to write marshalled data to

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.87.3.7 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalString (const std::string *value*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [protected, virtual]

Loose Marshal the String to the DataOuputStream passed.

Parameters:

value - string to **marshal** (p. 76)

dataOut - stream to write marshaled form to

Exceptions:

IOException if an error occurs.

6.87.3.8 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataInputStream *dis *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Loose Un-Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Un-Marshal

dis - the DataInputStream to Un-Marshal from

Exceptions:

IOException if an error occurs.

6.87.3.9 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalBroker (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unarshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshalled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.10 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalByteA (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal an array of char.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

Returns:

the unmarshalled vector of chars.

Exceptions:

IOException if an error occurs.

6.87.3.11 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalCache (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal the cached object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.12 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalConst (decaf::io::DataInputStream * *dataIn*, int *size*) throw (decaf::io::IOException) [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

size - size of the const array to unmarshal

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.87.3.13 virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalLong (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose **marshal** (p. 76) the long long type.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns:

the unmarshaled long long

Exceptions:

IOException if an error occurs.

6.87.3.14 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalNeste (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Unmarshal the nested object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.15 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseUnmarshalString (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Loose Un-Marshal the String to the DataOuputStream passed.

Parameters:

dataIn - stream to read marshaled form from

Returns:

the unmarshaled string

Exceptions:

IOException if an error occurs.

6.87.3.16 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::readAsciiString (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [protected, virtual]

Given an DataInputStream read a know ASCII formatted string from the input and return that string.

Parameters:

dataIn - DataInputStream to read from

Returns:

string value read from stream

6.87.3.17 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal1 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.18 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshal2 (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataOutputStream *ds *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.19 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerE (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

6.87.3.20 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalBrokerE (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshal the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

data - Error to Marshal

dataOut - stream to write marshalled data to

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.21 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedC (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters:

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to **marshal** (p. 76)

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.87.3.22 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalCachedC (OpenWireFormat * *wireFormat*, commands::DataStructure * *data*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters:

wireFormat - The OpenwireFormat properties

data - DataStructure Object Pointer to **marshal** (p. 76)

bs - boolean stream to **marshal** (p. 76) to.

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.87.3.23 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong1 (OpenWireFormat * *wireFormat*, long long *value*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly **marshal** (p. 76) the long long to the BooleanStream passed.

Parameters:

wireFormat - The OpenwireFormat properties

value - long long to **marshal** (p. 76)

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.87.3.24 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalLong2 (OpenWireFormat * *wireFormat*, long long *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly **marshal** (p. 76) the long long to the Streams passed.

Parameters:

wireFormat - The OpenwireFormat properties

value - long long to **marshal** (p. 76)

dataOut - stream to write marshaled form to

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.25 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedO (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed BooleanStream returning the size of the data marshaled.

Parameters:

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to **marshal** (p. 76)

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of data written.

Exceptions:

IOException if an error occurs.

6.87.3.26 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalNestedO (OpenWireFormat * *wireFormat*, commands::DataStructure * *object*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tightly marshals the passed DataStructure based object to the passed streams returning nothing.

Parameters:

wireFormat - The OpenwireFormat properties

object - DataStructure Object Pointer to **marshal** (p. 76)

bs - boolean stream to **marshal** (p. 76) to.

dataOut - stream to write marshaled data to

Exceptions:

IOException if an error occurs.

6.87.3.27 template<typename T > int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectA (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [inline, protected]

Tightly Marshal an array of DataStructure objects to the provided boolean stream, and return the size that the tight marshalling is going to take.

Parameters:

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.87.3.28 template<typename T > void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectA (OpenWireFormat * *wireFormat*, std::vector< T > *objects*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [inline, protected]

Tightly Marshal an array of DataStructure objects to the provided boolean stream and data output stream.

Parameters:

wireFormat - The OpenwireFormat properties

objects - array of DataStructure object pointers.

dataOut - stream to write marshalled data to

bs - boolean stream to **marshal** (p. 76) to.

Returns:

size of the marshalled data

Exceptions:

IOException if an error occurs.

References AMQ_CATCH_EXCEPTION_CONVERT, AMQ_CATCH_RETHROW, and AMQ_CATCHALL_THROW.

6.87.3.29 virtual int activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString1 (const std::string & *value*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshals the String to a Booleans Stream Object, returns the marshaled size.

Parameters:

value - string to **marshal** (p. 76)

bs - BooleanStream to use.

Returns:

size of marshaled string.

Exceptions:

IOException if an error occurs.

6.87.3.30 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalString2 (const std::string & *value*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Marshals the passed string to the streams passed.

Parameters:

value - string to **marshal** (p. 76)

dataOut - the DataOutputStream to Marshal to

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

6.87.3.31 virtual void activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshal (OpenWireFormat *format *AMQCPP_UNUSED*, commands::DataStructure *command *AMQCPP_UNUSED*, decaf::io::DataInputStream *dis *AMQCPP_UNUSED*, utils::BooleanStream *bs *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Tight Un-Marshal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Un-Marshal

dis - the DataInputStream to Un-Marshal from

bs - boolean stream to Un-Marshal from.

Exceptions:

IOException if an error occurs.

6.87.3.32 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalBroke (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Unarshall the Error object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshalled form from

bs - boolean stream to **marshal** (p. 76) to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.33 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalByteA (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Unmarshal an array of char.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

bs - boolean stream to unmarshal from.

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.87.3.34 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalCache (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Unmarshal the cached object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to **marshal** (p. 76) to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.35 virtual std::vector<unsigned char> activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalConst (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*, int *size*) throw (decaf::io::IOException) [protected, virtual]

Tight Unmarshal a fixed size array from that data input stream and return an stl vector of char as the resultant.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

bs - boolean stream to unmarshal from.

size - size of the const array to unmarshal

Returns:

the unmarshaled vector of chars.

Exceptions:

IOException if an error occurs.

6.87.3.36 virtual long long activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalLong (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight **marshal** (p. 76) the long long type.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to **marshal** (p. 76) to.

Returns:

the unmarshaled long long

Exceptions:

IOException if an error occurs.

6.87.3.37 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalNeste (OpenWireFormat * *wireFormat*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Tight Unmarshal the nested object.

Parameters:

wireFormat - The OpenwireFormat properties

dataIn - stream to read marshaled form from

bs - boolean stream to **marshal** (p. 76) to.

Returns:

pointer to a new DataStructure Object

Exceptions:

IOException if an error occurs.

6.87.3.38 virtual std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightUnmarshalString (decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [protected, virtual]

Performs Tight Unmarshaling of String Objects.

Parameters:

dataIn - the DataInputStream to Un-Marshal from

bs - boolean stream to unmarshal from.

Returns:

the unmarshaled string.

Exceptions:

IOException if an error occurs.

6.87.3.39 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toHexFromBytes (const std::vector< unsigned char > & *data*) [static]

given an array of bytes, convert that array to a Hexidecimal coded string that represents that data.

Parameters:

data - unsigned char data array pointer

Returns:

a string coded in hex that represents the data

6.87.3.40 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::TransactionId * *txnId*) [static]

Converts the given transaction ID into a String.

Parameters:

txnId - TransactionId poitner

Returns:

string representation of the id

6.87.3.41 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::ProducerId * *id*) [static]

Converts the object to a String.

Parameters:

id - ProducerId pointer

Returns:

string representing the id

6.87.3.42 static std::string activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::toString (const commands::MessageId * *id*) [static]

Converts the object to a String.

Parameters:

id - MessageId pointer

Returns:

string representing the id

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getCMSMessageID().

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**BaseDataStreamMarshaller.h**

6.88 activemq::commands::BaseDataStructure Class Reference

`#include <src/main/activemq/commands/BaseDataStructure.h>`Inheritance diagram for activemq::commands::BaseDataStructure:

Public Member Functions

- virtual ~**BaseDataStructure** ()
- virtual bool **isMarshalAware** () const

  Determine if this object is aware of marshaling and should have its before and after marshaling methods called.

- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Perform any processing needed before an marshal.

- virtual void **afterMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Perform any processing needed after an unmarshal.

- virtual void **beforeUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Perform any processing needed before an unmarshal.

- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Perform any processing needed after an unmarshal.

- virtual void **setMarshaledForm** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED, const std::vector< char > &data AMQCPP_UNUSED)

  Called to set the data to this object that will contain the objects marshaled form.

- virtual std::vector< unsigned char > **getMarshaledForm** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED)

  Called to get the data to this object that will contain the objects marshaled form.

- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value AMQCPP_UNUSED) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

6.88.1 Constructor & Destructor Documentation

6.88.1.1 virtual activemq::commands::BaseDataStructure::~BaseDataStructure () [inline, virtual]

6.88.2 Member Function Documentation

6.88.2.1 virtual void activemq::commands::BaseDataStructure::afterMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters:

wireformat (p. 74) - the OpenWireFormat object in use.

6.88.2.2 virtual void activemq::commands::BaseDataStructure::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed after an unmarshal.

Parameters:

wireformat (p. 74) - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 1741), and **activemq::commands::WireFormatInfo** (p. 2701).

6.88.2.3 virtual void activemq::commands::BaseDataStructure::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed before an marshal.

Parameters:

wireformat (p. 74) - the OpenWireFormat object in use.

Reimplemented in **activemq::commands::Message** (p. 1741), and **activemq::commands::WireFormatInfo** (p. 2702).

6.88.2.4 virtual void activemq::commands::BaseDataStructure::beforeUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

Perform any processing needed before an unmarshal.

Parameters:

wireformat (p. 74) - the OpenWireFormat object in use.

6.88.2.5 virtual void activemq::commands::BaseDataStructure::copyDataStructure (const DataStructure *src *AMQCPP_UNUSED*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented in **activemq::commands::BooleanExpression** (p. 579).

6.88.2.6 virtual bool activemq::commands::BaseDataStructure::equals (const DataStructure *value *AMQCPP_UNUSED*) const [inline, virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Referenced by activemq::commands::BooleanExpression::equals(), and activemq::commands::BaseCommand::equals().

6.88.2.7 virtual std::vector<unsigned char> activemq::commands::BaseDataStructure::getMarshaledForm (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

Returns:

buffer that holds the objects data.

6.88.2.8 virtual bool activemq::commands::BaseDataStructure::isMarshalAware () const [inline, virtual]

Determine if this object is aware of marshaling and should have its before and after marshaling methods called. Defaults to false.

Returns:

true if aware of marshaling

Implements **activemq::wireformat::MarshalAware** (p. 1714).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 262), **activemq::commands::ActiveMQStreamMessage** (p. 378), **activemq::commands::Message** (p. 1746), and **activemq::commands::WireFormatInfo** (p. 2704).

6.88.2.9 virtual void activemq::commands::BaseDataStructure::setMarshaledForm (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*, const std::vector< char > &data *AMQCPP_UNUSED*) [inline, virtual]

Called to set the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

data - vector of object binary data

6.88.2.10 virtual std::string activemq::commands::BaseDataStructure::toString () const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implements **activemq::commands::DataStructure** (p. 1177).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 150), **activemq::commands::ActiveMQBytesMessage** (p. 173), **activemq::commands::ActiveMQDestination** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 266), **activemq::commands::ActiveMQMessage** (p. 281), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQQueue** (p. 340), **activemq::commands::ActiveMQStreamMessage** (p. 382), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), **activemq::commands::ActiveMQTextMessage** (p. 452), **activemq::commands::ActiveMQTopic** (p. 468), **activemq::commands::BaseCommand** (p. 512), **activemq::commands::BooleanExpression** (p. 579), **activemq::commands::BrokerId** (p. 594), **activemq::commands::BrokerInfo** (p. 612), **activemq::commands::Command** (p. 883), **activemq::commands::ConnectionControl** (p. 948), **activemq::commands::ConnectionError** (p. 964), **activemq::commands::ConnectionId** (p. 983), **activemq::commands::ConnectionInfo** (p. 1001), **activemq::commands::ConsumerControl** (p. 1031), **activemq::commands::ConsumerId** (p. 1048), **activemq::commands::ConsumerInfo** (p. 1067), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::DestinationInfo** (p. 1197), **activemq::commands::DiscoveryEvent** (p. 1215), **activemq::commands::ExceptionResponse** (p. 1278), **activemq::commands::FlushCommand** (p. 1359), **activemq::commands::IntegerResponse** (p. 1444), **activemq::commands::JournalQueueAck** (p. 1493), **activemq::commands::JournalTopicAck** (p. 1510), **activemq::commands::JournalTrace** (p. 1526), **activemq::commands::JournalTransaction** (p. 1541), **activemq::commands::KeepAliveInfo** (p. 1557), **activemq::commands::LastPartialCommand** (p. 1577), **activemq::commands::LocalTransactionId** (p. 1603), **activemq::commands::Message** (p. 1749), **activemq::commands::MessageAck**

(p. 1781), **activemq::commands::MessageDispatch** (p. 1803), **activemq::commands::MessageDispatchNotification** (p. 1827), **activemq::commands::MessageId** (p. 1846), **activemq::commands::MessagePull** (p. 1896), **activemq::commands::NetworkBridgeFilter** (p. 1926), **activemq::commands::PartialCommand** (p. 1997), **activemq::commands::ProducerAck** (p. 2088), **activemq::commands::ProducerId** (p. 2108), **activemq::commands::ProducerInfo** (p. 2125), **activemq::commands::RemoveInfo** (p. 2179), **activemq::commands::RemoveSubscriptionInfo** (p. 2196), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2233), **activemq::commands::SessionId** (p. 2287), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2351), **activemq::commands::SubscriptionInfo** (p. 2494), **activemq::commands::TransactionId** (p. 2575), **activemq::commands::TransactionInfo** (p. 2592), **activemq::commands::WireFormatInfo** (p. 2708), and **activemq::commands::XATransactionId** (p. 2734).

Referenced by activemq::commands::BooleanExpression::toString(), and activemq::commands::BaseCommand::toString().

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BaseDataStructure.h**

6.89 binary_function Class Reference

Inheritance diagram for binary_function:

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Comparator.h**

6.90 decaf::net::BindException Class Reference

`#include <src/main/decaf/net/BindException.h>`Inheritance diagram for decaf::net::BindException:

Public Member Functions

- **BindException** () throw ()

  Default Constructor.

- **BindException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **BindException** (const **BindException** &ex) throw ()

  Copy Constructor.

- **BindException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **BindException** (const std::exception ***cause**) throw ()

  Constructor.

- **BindException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual **BindException** * **clone** () const

  Clones this exception.

- virtual ~**BindException** () throw ()

6.90.1 Constructor & Destructor Documentation

6.90.1.1 decaf::net::BindException::BindException () throw () `[inline]`

Default Constructor.

6.90.1.2 decaf::net::BindException::BindException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.90.1.3 decaf::net::BindException::BindException (const BindException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.90.1.4 decaf::net::BindException::BindException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.90.1.5 decaf::net::BindException::BindException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.90.1.6 decaf::net::BindException::BindException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.90.1.7 virtual decaf::net::BindException::~BindException () throw () [inline, virtual]

6.90.2 Member Function Documentation

6.90.2.1 virtual BindException* decaf::net::BindException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2380).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**BindException.h**

6.91 decaf::io::BlockingByteArrayInputStream Class Reference

This is a blocking version of a byte buffer stream.

`#include <src/main/decaf/io/BlockingByteArrayInputStream.h>`Inheritance diagram for decaf::io::BlockingByteArrayInputStream:

Public Member Functions

- **BlockingByteArrayInputStream** ()

  *Default Constructor - uses a default **internal** (p. 95) buffer.*

- **BlockingByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)

  *Constructor that initializes the **internal** (p. 95) buffer.*

- virtual ~**BlockingByteArrayInputStream** ()

  Destructor.

- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)

  Sets the data that this reader uses.

- virtual std::size_t **available** () const throw (IOException)

  Indicates the number of bytes available to be read without blocking.

- virtual unsigned char **read** () throw (IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual void **close** () throw (lang::Exception)

  Closes the target input stream.

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit DECAF_UNUSED)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

Determines if this input stream supports the mark and reset methods.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.91.1 Detailed Description

This is a blocking version of a byte buffer stream. Read operations block until the requested data becomes available in the **internal** (p. 95) buffer via a call to setByteArray.

6.91.2 Constructor & Destructor Documentation

6.91.2.1 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream ()

Default Constructor - uses a default **internal** (p. 95) buffer.

6.91.2.2 decaf::io::BlockingByteArrayInputStream::BlockingByteArrayInputStream (const unsigned char * *buffer*, std::size_t *bufferSize*)

Constructor that initializes the **internal** (p. 95) buffer.

See also:

setByteArray (p. 570).

6.91.2.3 virtual decaf::io::BlockingByteArrayInputStream::~BlockingByteArrayInputStream () [virtual]

Destructor.

6.91.3 Member Function Documentation

6.91.3.1 virtual std::size_t decaf::io::BlockingByteArrayInputStream::available () const throw (IOException) [inline, virtual]

Indicates the number of bytes available to be read without blocking.

Returns:

the data available in the **internal** (p. 95) buffer.

Exceptions:

IOException *(p. 1477)* if an error occurs.

Implements **decaf::io::InputStream** (p. 1407).

6.91.3.2 virtual void decaf::io::BlockingByteArrayInputStream::close () throw (lang::Exception) [virtual]

Closes the target input stream.

Exceptions:

IOException *(p. 1477)* if an error occurs.

Implements **decaf::io::Closeable** (p. 840).

6.91.3.3 virtual void decaf::io::BlockingByteArrayInputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.91.3.4 virtual void decaf::io::BlockingByteArrayInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1407).

6.91.3.5 virtual bool decaf::io::BlockingByteArrayInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1407).

6.91.3.6 virtual void decaf::io::BlockingByteArrayInputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.91.3.7 virtual void decaf::io::BlockingByteArrayInputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.91.3.8 virtual int decaf::io::BlockingByteArrayInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available in the buffer via a call to setByteArray.

Parameters:

buffer (out) the target buffer

offset the position in the buffer to start from.

bufferSize the size of the output buffer.

Returns:

the number of bytes read. or -1 if EOF

Exceptions:

IOException *(*p. ***1477***)* f an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.91.3.9 virtual unsigned char decaf::io::BlockingByteArrayInputStream::read () throw (IOException) [virtual]

Reads a single byte from the buffer. This operation will block until data has been added to the buffer via a call to setByteArray.

Returns:

the next byte.

Exceptions:

IOException *(*p. ***1477***)* if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.91.3.10 virtual void decaf::io::BlockingByteArrayInputStream::reset () throw (IOException) [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: ∗ If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1477) might be thrown. ∗ If such an **IOException** (p. 1477) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: ∗ The call to reset may throw an **IOException** (p. 1477). ∗ If an **IOException** (p. 1477) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException *(*p. ***1477***)*

Implements **decaf::io::InputStream** (p. 1408).

6.91.3.11 virtual void decaf::io::BlockingByteArrayInputStream::setByteArray (const unsigned char ∗ *buffer*, std::size_t *bufferSize*) [virtual]

Sets the data that this reader uses. Replaces any existing data and resets the read index to the beginning of the buffer. When this method is called, it notifies any other threads that data is now available to be read.

Parameters:

buffer The new data to be copied to the **internal** (p. 95) buffer.

bufferSize The size of the new buffer.

6.91.3.12 virtual std::size_t decaf::io::BlockingByteArrayInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1406) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total butes skipped

Exceptions:

IOException *(*p. *1477)* if an error occurs

Implements **decaf::io::InputStream** (p. 1409).

6.91.3.13 virtual void decaf::io::BlockingByteArrayInputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.91.3.14 virtual void decaf::io::BlockingByteArrayInputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.91.3.15 virtual void decaf::io::BlockingByteArrayInputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BlockingByteArrayInputStream.h**

6.92 decaf::lang::Boolean Class Reference

`#include <src/main/decaf/lang/Boolean.h>`Inheritance diagram for decaf::lang::Boolean:

Public Member Functions

- **Boolean** (bool value)
- **Boolean** (const std::string &value)
- virtual ~**Boolean** ()
- bool **booleanValue** () const
- std::string **toString** () const
- virtual int **compareTo** (const **Boolean** &b) const

  *Compares this **Boolean** (p. 573) instance with another.*

- virtual bool **operator==** (const **Boolean** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Boolean** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- bool **equals** (const **Boolean** &b) const
- virtual int **compareTo** (const bool &b) const

  *Compares this **Boolean** (p. 573) instance with another.*

- virtual bool **operator==** (const bool &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const bool &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- bool **equals** (const bool &b) const

Static Public Member Functions

- static **Boolean valueOf** (bool value)
- static **Boolean valueOf** (const std::string &value)
- static bool **parseBoolean** (const std::string &value)

  Parses the String passed and extracts an bool.

- static std::string **toString** (bool value)

  Converts the bool to a String representation.

Static Public Attributes

- static const **Boolean _FALSE**

  The Class object representing the primitive false boolean.

- static const **Boolean _TRUE**

  The Class object representing the primitive type boolean.

6.92.1 Constructor & Destructor Documentation

6.92.1.1 decaf::lang::Boolean::Boolean (bool *value*)

Parameters:

value - primitive boolean to wrap.

6.92.1.2 decaf::lang::Boolean::Boolean (const std::string & *value*)

Parameters:

value - String value to convert to a boolean.

6.92.1.3 virtual decaf::lang::Boolean::~Boolean () [inline, virtual]

6.92.2 Member Function Documentation

6.92.2.1 bool decaf::lang::Boolean::booleanValue () const [inline]

Returns:

the primitive boolean value of this object

6.92.2.2 virtual int decaf::lang::Boolean::compareTo (const bool & *b*) const [virtual]

Compares this **Boolean** (p. 573) instance with another.

Parameters:

b - the **Boolean** (p. 573) instance to be compared

Returns:

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

Implements **decaf::lang::Comparable< bool >** (p. 899).

6.92.2.3 virtual int decaf::lang::Boolean::compareTo (const Boolean & *b*) const [virtual]

Compares this **Boolean** (p. 573) instance with another.

Parameters:

b - the **Boolean** (p. 573) instance to be compared

Returns:

zero if this object represents the same boolean value as the argument; a positive value if this object represents true and the argument represents false; and a negative value if this object represents false and the argument represents true

6.92.2.4 bool decaf::lang::Boolean::equals (const bool & *b*) const [inline, virtual]

Returns:

true if the two **Boolean** (p. 573) Objects have the same value.

Implements **decaf::lang::Comparable< bool >** (p. 900).

6.92.2.5 bool decaf::lang::Boolean::equals (const Boolean & *b*) const [inline]

Returns:

true if the two **Boolean** (p. 573) Objects have the same value.

6.92.2.6 virtual bool decaf::lang::Boolean::operator< (const bool & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< bool >** (p. 900).

6.92.2.7 virtual bool decaf::lang::Boolean::operator< (const Boolean & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.92.2.8 virtual bool decaf::lang::Boolean::operator== (const bool & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< bool >** (p. 901).

6.92.2.9 virtual bool decaf::lang::Boolean::operator== (const Boolean & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.92.2.10 static bool decaf::lang::Boolean::parseBoolean (const std::string & *value*) [static]

Parses the String passed and extracts an bool.

Parameters:

value The std::string value to parse

Returns:

bool value

6.92.2.11 static std::string decaf::lang::Boolean::toString (bool *value*) [static]

Converts the bool to a String representation.

Parameters:

value The bool value to convert.

Returns:

std::string representation of the bool value passed.

6.92.2.12 std::string decaf::lang::Boolean::toString () const

Returns:

the string representation of this Booleans value.

6.92.2.13 static Boolean decaf::lang::Boolean::valueOf (const std::string & *value*) [static]

Parameters:

value The std::string value to convert to a `Boolean` (p. 573) instance.

Returns:

a **Boolean** (p. 573) instance of the string value

6.92.2.14 static Boolean decaf::lang::Boolean::valueOf (bool *value*) [static]

Parameters:

value The bool value to convert to a `Boolean` (p. 573) instance.

Returns:

a **Boolean** (p. 573) instance of the primitive boolean value

6.92.3 Field Documentation

6.92.3.1 const Boolean decaf::lang::Boolean::_FALSE [static]

The Class object representing the primitive false boolean.

6.92.3.2 const Boolean decaf::lang::Boolean::_TRUE [static]

The Class object representing the primitive type boolean.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Boolean.h**

6.93 activemq::commands::BooleanExpression Class Reference

`#include <src/main/activemq/commands/BooleanExpression.h>`Inheritance diagram for activemq::commands::BooleanExpression:

Public Member Functions

- **BooleanExpression** ()
- virtual ~**BooleanExpression** ()
- virtual **DataStructure** * **cloneDataStructure** () const

  Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src AMQCPP_UNUSED)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

6.93.1 Constructor & Destructor Documentation

6.93.1.1 activemq::commands::BooleanExpression::BooleanExpression () [inline]

6.93.1.2 virtual activemq::commands::BooleanExpression::~BooleanExpression () [inline, virtual]

6.93.2 Member Function Documentation

6.93.2.1 virtual DataStructure* activemq::commands::BooleanExpression::cloneDataStructure () const [inline, virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.93.2.2 virtual void activemq::commands::BooleanExpression::copyDataStructure (const DataStructure *src *AMQCPP_UNUSED*) [inline, virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::BaseDataStructure** (p. 559).

6.93.2.3 virtual bool activemq::commands::BooleanExpression::equals (const DataStructure * *value*) const [inline, virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

References activemq::commands::BaseDataStructure::equals().

6.93.2.4 virtual std::string activemq::commands::BooleanExpression::toString () const [inline, virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

References activemq::commands::BaseDataStructure::toString().

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BooleanExpression.h**

6.94 activemq::wireformat::openwire::utils::BooleanStream Class Reference

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

`#include <src/main/activemq/wireformat/openwire/utils/BooleanStream.h>`

Public Member Functions

- **BooleanStream** ()
- virtual ~**BooleanStream** ()
- bool **readBoolean** () throw (decaf::io::IOException)

  Read a boolean data element from the internal data buffer.

- void **writeBoolean** (bool value) throw (decaf::io::IOException)

  Writes a Boolean value to the internal data buffer.

- void **marshal** (**decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Marshal the data to a DataOutputStream.

- void **marshal** (std::vector< unsigned char > &dataOut)

  Marshal the data to a STL vector of unsigned chars.

- void **unmarshal** (**decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Unmarshal a Boolean data stream from the Input Stream.

- void **clear** ()

  Clears to old position markers, data starts at the beginning.

- int **marshalledSize** ()

  Calc the size that data is marshalled to.

6.94.1 Detailed Description

Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream. The booleans are stored as single bits in the stream, with the stream size pre-pended to the stream when the data is marshalled.

The serialized form of the size field can be between 1 and 3 bytes. If the number of used bytes < 64, size=1 byte If the number of used bytes >=64 and < 256 (size of an unsigned byte), size=2 bytes If the number of used bytes >=256, size=3 bytes

The high-order 2 bits (128 and 64) of the first byte of the size field are used to encode the information about the number of bytes in the size field. The only time the first byte will contain a value >=64 is if there are more bytes in the size field. If the first byte < 64, the value of the byte is simply the size value. If the first byte = 0xC0, the following unsigned byte is the size field. If the first byte = 0x80, the following short (two bytes) are the size field.

6.94.2 Constructor & Destructor Documentation

6.94.2.1 activemq::wireformat::openwire::utils::BooleanStream::BooleanStream ()

6.94.2.2 virtual activemq::wireformat::openwire::utils::BooleanStream::~BooleanStream () [inline, virtual]

6.94.3 Member Function Documentation

6.94.3.1 void activemq::wireformat::openwire::utils::BooleanStream::clear ()

Clears to old position markers, data starts at the beginning.

6.94.3.2 void activemq::wireformat::openwire::utils::BooleanStream::marshal (std::vector< unsigned char > & *dataOut*)

Marshal the data to a STL vector of unsigned chars.

Parameters:

dataOut - reference to a vector to write the data to.

6.94.3.3 void activemq::wireformat::openwire::utils::BooleanStream::marshal (decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)

Marshal the data to a DataOutputStream.

Parameters:

dataOut - Stream to write the data to.

6.94.3.4 int activemq::wireformat::openwire::utils::BooleanStream::marshalledSize ()

Calc the size that data is marshalled to.

Returns:

int size of marshalled data.

6.94.3.5 bool activemq::wireformat::openwire::utils::BooleanStream::readBoolean () throw (decaf::io::IOException)

Read a boolean data element from the internal data buffer.

Returns:

boolean from the stream

6.94.3.6 void activemq::wireformat::openwire::utils::BooleanStream::unmarshal (decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException)

Unmarshal a Boolean data stream from the Input Stream.

Parameters:

dataIn - Input Stream to read data from.

6.94.3.7 void activemq::wireformat::openwire::utils::BooleanStream::writeBoolean (bool *value*) throw (decaf::io::IOException)

Writes a Boolean value to the internal data buffer.

Parameters:

value - boolean data to write.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**BooleanStream.h**

6.95 decaf::util::concurrent::BrokenBarrierException Class Reference

`#include <src/main/decaf/util/concurrent/BrokenBarrierException.h>`Inheritance diagram for decaf::util::concurrent::BrokenBarrierException:

Public Member Functions

- **BrokenBarrierException** () throw ()
  Default Constructor.
- **BrokenBarrierException** (const **decaf::lang::Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **BrokenBarrierException** (const **BrokenBarrierException** &ex) throw ()
  Copy Constructor.
- **BrokenBarrierException** (const std::exception ∗**cause**) throw ()
  Constructor.
- **BrokenBarrierException** (const char ∗file, const int lineNumber, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **BrokenBarrierException** (const char ∗file, const int lineNumber, const std::exception ∗**cause**, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **BrokenBarrierException** ∗ **clone** () const
  Clones this exception.
- virtual ∼**BrokenBarrierException** () throw ()

6.95.1 Constructor & Destructor Documentation

6.95.1.1 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException () throw () `[inline]`

Default Constructor.

6.95.1.2 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const decaf::lang::Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.95.1.3 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const BrokenBarrierException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.95.1.4 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.95.1.5 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.95.1.6 decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.95.1.7 virtual decaf::util::concurrent::BrokenBarrierException::~BrokenBarrierException () throw () [inline, virtual]

6.95.2 Member Function Documentation

6.95.2.1 virtual BrokenBarrierException* decaf::util::concurrent::BrokenBarrierException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**BrokenBarrierException.h**

6.96 activemq::commands::BrokerError Class Reference

This class represents an Exception sent from the Broker.

`#include <src/main/activemq/commands/BrokerError.h>`Inheritance diagram for activemq::commands::BrokerError:

Data Structures

- struct **StackTraceElement**

Public Member Functions

- **BrokerError** ()
- virtual ~**BrokerError** ()
- virtual unsigned char **getDataStructureType** () const

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **BrokerError** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

- virtual const std::string & **getMessage** () const

  Gets the string holding the error message.

- virtual void **setMessage** (const std::string &message)

  *Sets the string that contains the error **Message** (p. 1737).*

- virtual const std::string & **getExceptionClass** () const

  Gets the string holding the Exception Class name.

- virtual void **setExceptionClass** (const std::string &exceptionClass)

  Sets the string that contains the Exception Class name.

- virtual const **decaf::lang::Pointer**< **BrokerError** > & **getCause** () const

  Gets the Broker Error that caused this exception.

- virtual void **setCause** (const **decaf::lang::Pointer**< **BrokerError** > &cause)

  Sets the Broker Error that caused this exception.

- virtual const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > & **getStackTraceElements** () const

  Gets the Stack Trace Elements for the Exception.

- virtual void **setStackTraceElements** (const std::vector< **decaf::lang::Pointer**< **StackTraceElement** > > &stackTraceElements)

  Sets the Stack Trace Elements for this Exception.

6.96.1 Detailed Description

This class represents an Exception sent from the Broker. The Broker sends java Throwables, so we must mimic its structure here.

6.96.2 Constructor & Destructor Documentation

6.96.2.1 activemq::commands::BrokerError::BrokerError () [inline]

6.96.2.2 virtual activemq::commands::BrokerError::~BrokerError () [virtual]

6.96.3 Member Function Documentation

6.96.3.1 virtual BrokerError* activemq::commands::BrokerError::cloneDataStructure () const [inline, virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

References copyDataStructure().

6.96.3.2 virtual void activemq::commands::BrokerError::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

Referenced by cloneDataStructure().

6.96.3.3 virtual const decaf::lang::Pointer<BrokerError>& activemq::commands::BrokerError::getCause () const [inline, virtual]

Gets the Broker Error that caused this exception.

Returns:

Broker Error Pointer

6.96.3.4 virtual unsigned char activemq::commands::BrokerError::getDataStructureType () const [inline, virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implements **activemq::commands::DataStructure** (p. 1176).

6.96.3.5 virtual const std::string& activemq::commands::BrokerError::getExceptionClass () const [inline, virtual]

Gets the string holding the Exception Class name.

Returns:

Exception Class name

6.96.3.6 virtual const std::string& activemq::commands::BrokerError::getMessage () const [inline, virtual]

Gets the string holding the error message.

Returns:

String **Message** (p. 1737)

6.96.3.7 virtual const std::vector< decaf::lang::Pointer<StackTraceElement> >& activemq::commands::BrokerError::getStackTraceElements () const [inline, virtual]

Gets the Stack Trace Elements for the Exception.

Returns:

Stack Trace Elements

6.96.3.8 virtual void activemq::commands::BrokerError::setCause (const decaf::lang::Pointer< BrokerError > & *cause*) [inline, virtual]

Sets the Broker Error that caused this exception.

Parameters:

cause - Broker Error

6.96.3.9 virtual void activemq::commands::BrokerError::setExceptionClass (const std::string & *exceptionClass*) [inline, virtual]

Sets the string that contains the Exception Class name.

Parameters:

exceptionClass - String Exception Class name

6.96.3.10 virtual void activemq::commands::BrokerError::setMessage (const std::string & *message*) [inline, virtual]

Sets the string that contains the error **Message** (p. 1737).

Parameters:

message - String Error **Message** (p. 1737)

6.96.3.11 virtual void activemq::commands::BrokerError::setStackTraceElements (const std::vector< decaf::lang::Pointer< StackTraceElement > > & *stackTraceElements*) [inline, virtual]

Sets the Stack Trace Elements for this Exception.

Parameters:

stackTraceElements - Stack Trace Elements

6.96.3.12 virtual decaf::lang::Pointer<commands::Command> activemq::commands::BrokerError::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.97 activemq::exceptions::BrokerException Class Reference

`#include <src/main/activemq/exceptions/BrokerException.h>`Inheritance diagram for activemq::exceptions::BrokerException:

Public Member Functions

- **BrokerException** () throw ()
- **BrokerException** (const **exceptions::ActiveMQException** &ex) throw ()
- **BrokerException** (const **BrokerException** &ex) throw ()
- **BrokerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
- **BrokerException** (const char *file, const int lineNumber, const **commands::BrokerError** *error) throw ()
- virtual **BrokerException** * **clone** () const

  Clones this exception.

- virtual ~**BrokerException** () throw ()

6.97.1 Constructor & Destructor Documentation

6.97.1.1 activemq::exceptions::BrokerException::BrokerException () throw () [inline]

6.97.1.2 activemq::exceptions::BrokerException::BrokerException (const exceptions::ActiveMQException & *ex*) throw () [inline]

6.97.1.3 activemq::exceptions::BrokerException::BrokerException (const BrokerException & *ex*) throw () [inline]

6.97.1.4 activemq::exceptions::BrokerException::BrokerException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

6.97.1.5 activemq::exceptions::BrokerException::BrokerException (const char * *file*, const int *lineNumber*, const commands::BrokerError * *error*) throw () [inline]

References decaf::lang::Exception::setMark(), and decaf::lang::Exception::setMessage().

6.97.1.6 virtual activemq::exceptions::BrokerException::~BrokerException () throw () [inline, virtual]

6.97.2 Member Function Documentation

6.97.2.1 virtual BrokerException* activemq::exceptions::BrokerException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **activemq::exceptions::ActiveMQException** (p. 253).

The documentation for this class was generated from the following file:

- src/main/activemq/exceptions/**BrokerException.h**

6.98 activemq::commands::BrokerId Class Reference

`#include <src/main/activemq/commands/BrokerId.h>`Inheritance diagram for activemq::commands::BrokerId:

Public Types

- typedef **decaf::lang::PointerComparator< BrokerId > COMPARATOR**

Public Member Functions

- **BrokerId** ()
- **BrokerId** (const **BrokerId** &other)
- virtual ~**BrokerId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **BrokerId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &**value**)
- virtual int **compareTo** (const **BrokerId** &**value**) const
- virtual bool **equals** (const **BrokerId** &**value**) const
- virtual bool **operator==** (const **BrokerId** &**value**) const
- virtual bool **operator<** (const **BrokerId** &**value**) const
- **BrokerId** & **operator=** (const **BrokerId** &other)

Static Public Attributes

- static const unsigned char **ID_BROKERID** = 124

Protected Attributes

- std::string **value**

6.98.1 Member Typedef Documentation

6.98.1.1 typedef decaf::lang::PointerComparator<BrokerId> activemq::commands::BrokerId::COMPARATOR

6.98.2 Constructor & Destructor Documentation

6.98.2.1 activemq::commands::BrokerId::BrokerId ()

6.98.2.2 activemq::commands::BrokerId::BrokerId (const BrokerId & *other*)

6.98.2.3 virtual activemq::commands::BrokerId::~BrokerId () [virtual]

6.98.3 Member Function Documentation

6.98.3.1 virtual BrokerId* activemq::commands::BrokerId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.98.3.2 virtual int activemq::commands::BrokerId::compareTo (const BrokerId & *value*) const [virtual]

6.98.3.3 virtual void activemq::commands::BrokerId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.98.3.4 virtual bool activemq::commands::BrokerId::equals (const BrokerId & *value*) const [virtual]

6.98.3.5 virtual bool activemq::commands::BrokerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.98.3.6 virtual unsigned char activemq::commands::BrokerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.98.3.7 virtual std::string& activemq::commands::BrokerId::getValue () [virtual]

6.98.3.8 virtual const std::string& activemq::commands::BrokerId::getValue () const [virtual]

6.98.3.9 virtual bool activemq::commands::BrokerId::operator< (const BrokerId & *value*) const [virtual]

6.98.3.10 BrokerId& activemq::commands::BrokerId::operator= (const BrokerId & *other*)

6.98.3.11 virtual bool activemq::commands::BrokerId::operator== (const BrokerId & *value*) const [virtual]

6.98.3.12 virtual void activemq::commands::BrokerId::setValue (const std::string & *value*) [virtual]

6.98.3.13 virtual std::string activemq::commands::BrokerId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.98.4 Field Documentation

6.98.4.1 const unsigned char activemq::commands::BrokerId::ID_BROKERID = 124 [static]

6.98.4.2 std::string activemq::commands::BrokerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerId.h**

6.99 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 595).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual ~**BrokerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.99.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 595). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.99.2 Constructor & Destructor Documentation

6.99.2.1 activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::BrokerIdMarshaller () [inline]

6.99.2.2 virtual activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::~BrokerIdMarshaller () [inline, virtual]

6.99.3 Member Function Documentation

6.99.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.99.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.99.3.3 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.99.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.99.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.99.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.99.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerIdMarshaller.h**

6.100 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 599).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual ~**BrokerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.100.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 599). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.100.2 Constructor & Destructor Documentation

6.100.2.1 activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::BrokerIdMarshaller () [inline]

6.100.2.2 virtual activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::~BrokerIdMarshaller () [inline, virtual]

6.100.3 Member Function Documentation

6.100.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.100.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.100.3.3 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.100.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.100.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.100.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.100.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerIdMarshaller.h**

6.101 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller Class Reference

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 603).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller:

Public Member Functions

- **BrokerIdMarshaller** ()
- virtual ~**BrokerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.101.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 603). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.101.2 Constructor & Destructor Documentation

6.101.2.1 activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::BrokerIdMarshaller () [inline]

6.101.2.2 virtual activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::~BrokerIdMarshaller () [inline, virtual]

6.101.3 Member Function Documentation

6.101.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.101.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.101.3.3 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.101.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.101.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.101.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.101.3.7 virtual void activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerIdMarshaller.h**

6.102 activemq::commands::BrokerInfo Class Reference

`#include <src/main/activemq/commands/BrokerInfo.h>`Inheritance diagram for activemq::commands::BrokerInfo:

Public Member Functions

- **BrokerInfo** ()
- virtual ~**BrokerInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **BrokerInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **BrokerId** > & **getBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getBrokerId** ()
- virtual void **setBrokerId** (const **Pointer**< **BrokerId** > &**brokerId**)
- virtual const std::string & **getBrokerURL** () const
- virtual std::string & **getBrokerURL** ()
- virtual void **setBrokerURL** (const std::string &**brokerURL**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > & **getPeerBrokerInfos** ()
- virtual void **setPeerBrokerInfos** (const std::vector< **decaf::lang::Pointer**< **BrokerInfo** > > &**peerBrokerInfos**)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &**brokerName**)
- virtual bool **isSlaveBroker** () const
- virtual void **setSlaveBroker** (bool **slaveBroker**)
- virtual bool **isMasterBroker** () const
- virtual void **setMasterBroker** (bool **masterBroker**)
- virtual bool **isFaultTolerantConfiguration** () const
- virtual void **setFaultTolerantConfiguration** (bool **faultTolerantConfiguration**)
- virtual bool **isDuplexConnection** () const
- virtual void **setDuplexConnection** (bool **duplexConnection**)

- virtual bool **isNetworkConnection** () const
- virtual void **setNetworkConnection** (bool **networkConnection**)
- virtual long long **getConnectionId** () const
- virtual void **setConnectionId** (long long **connectionId**)
- virtual const std::string & **getBrokerUploadUrl** () const
- virtual std::string & **getBrokerUploadUrl** ()
- virtual void **setBrokerUploadUrl** (const std::string &**brokerUploadUrl**)
- virtual const std::string & **getNetworkProperties** () const
- virtual std::string & **getNetworkProperties** ()
- virtual void **setNetworkProperties** (const std::string &**networkProperties**)
- virtual bool **isBrokerInfo** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_BROKERINFO** = 2

Protected Member Functions

- **BrokerInfo** (const **BrokerInfo** &)
- **BrokerInfo** & **operator=** (const **BrokerInfo** &)

Protected Attributes

- **Pointer< BrokerId > brokerId**
- std::string **brokerURL**
- std::vector< **decaf::lang::Pointer< BrokerInfo > > peerBrokerInfos**
- std::string **brokerName**
- bool **slaveBroker**
- bool **masterBroker**
- bool **faultTolerantConfiguration**
- bool **duplexConnection**
- bool **networkConnection**
- long long **connectionId**
- std::string **brokerUploadUrl**
- std::string **networkProperties**

6.102.1 Constructor & Destructor Documentation

6.102.1.1 activemq::commands::BrokerInfo::BrokerInfo (const BrokerInfo &) [inline, protected]

6.102.1.2 activemq::commands::BrokerInfo::BrokerInfo ()

6.102.1.3 virtual activemq::commands::BrokerInfo::~BrokerInfo () [virtual]

6.102.2 Member Function Documentation

6.102.2.1 virtual BrokerInfo* activemq::commands::BrokerInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.102.2.2 virtual void activemq::commands::BrokerInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.102.2.3 virtual bool activemq::commands::BrokerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.102.2.4 virtual Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () [virtual]

6.102.2.5 virtual const Pointer<BrokerId>& activemq::commands::BrokerInfo::getBrokerId () const [virtual]

6.102.2.6 virtual std::string& activemq::commands::BrokerInfo::getBrokerName () [virtual]

6.102.2.7 virtual const std::string& activemq::commands::BrokerInfo::getBrokerName () const [virtual]

6.102.2.8 virtual std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () [virtual]

6.102.2.9 virtual const std::string& activemq::commands::BrokerInfo::getBrokerUploadUrl () const [virtual]

6.102.2.10 virtual std::string& activemq::commands::BrokerInfo::getBrokerURL () [virtual]

6.102.2.11 virtual const std::string& activemq::commands::BrokerInfo::getBrokerURL () const [virtual]

6.102.2.12 virtual long long activemq::commands::BrokerInfo::getConnectionId () const [virtual]

6.102.2.13 virtual unsigned char activemq::commands::BrokerInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.102.2.14 virtual std::string& activemq::commands::BrokerInfo::getNetworkProperties () [virtual]

6.102.2.15 virtual const std::string& activemq::commands::BrokerInfo::getNetworkProperties () const [virtual]

6.102.2.16 virtual std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () [virtual]

6.102.2.17 virtual const std::vector< decaf::lang::Pointer<BrokerInfo> >& activemq::commands::BrokerInfo::getPeerBrokerInfos () const [virtual]

6.102.2.18 virtual bool activemq::commands::BrokerInfo::isBrokerInfo () const [inline, virtual]

Returns:

an answer of true to the **isBrokerInfo()** (p. 611) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 509).

6.102.2.19 virtual bool activemq::commands::BrokerInfo::isDuplexConnection () const [virtual]

6.102.2.20 virtual bool activemq::commands::BrokerInfo::isFaultTolerantConfiguration () const [virtual]

6.102.2.21 virtual bool activemq::commands::BrokerInfo::isMasterBroker () const [virtual]

6.102.2.22 virtual bool activemq::commands::BrokerInfo::isNetworkConnection () const [virtual]

6.102.2.23 virtual bool activemq::commands::BrokerInfo::isSlaveBroker () const [virtual]

6.102.2.24 BrokerInfo& activemq::commands::BrokerInfo::operator= (const BrokerInfo &) [inline, protected]

6.102.2.25 virtual void activemq::commands::BrokerInfo::setBrokerId (const Pointer< BrokerId > & *brokerId*) [virtual]

6.102.2.26 virtual void activemq::commands::BrokerInfo::setBrokerName (const std::string & *brokerName*) [virtual]

6.102.2.27 virtual void activemq::commands::BrokerInfo::setBrokerUploadUrl (const std::string & *brokerUploadUrl*) [virtual]

6.102.2.28 virtual void activemq::commands::BrokerInfo::setBrokerURL (const std::string & *brokerURL*) [virtual]

6.102.2.29 virtual void activemq::commands::BrokerInfo::setConnectionId (long long *connectionId*) [virtual]

6.102.2.30 virtual void activemq::commands::BrokerInfo::setDuplexConnection (bool *duplexConnection*) [virtual]

6.102.2.31 virtual void activemq::commands::BrokerInfo::setFaultTolerantConfiguration (bool *faultTolerantConfiguration*) [virtual]

6.102.2.32 virtual void activemq::commands::BrokerInfo::setMasterBroker (bool *masterBroker*) [virtual]

6.102.2.33 virtual void activemq::commands::BrokerInfo::setNetworkConnection (bool *networkConnection*) [virtual]

6.102.2.34 virtual void activemq::commands::BrokerInfo::setNetworkProperties (const std::string & *networkProperties*) [virtual]

6.102.2.35 virtual void activemq::commands::BrokerInfo::setPeerBrokerInfos (const std::vector< decaf::lang::Pointer< BrokerInfo > > & *peerBrokerInfos*) [virtual]

6.102.2.36 virtual void activemq::commands::BrokerInfo::setSlaveBroker (bool *slaveBroker*) [virtual]

6.102.2.37 virtual std::string activemq::commands::BrokerInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.102.2.38 virtual Pointer<Command> activemq::commands::BrokerInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.102.3 Field Documentation

6.102.3.1 Pointer<BrokerId> activemq::commands::BrokerInfo::brokerId [protected]

6.102.3.2 std::string activemq::commands::BrokerInfo::brokerName [protected]

6.102.3.3 std::string activemq::commands::BrokerInfo::brokerUploadUrl [protected]

6.102.3.4 std::string activemq::commands::BrokerInfo::brokerURL [protected]

6.102.3.5 long long activemq::commands::BrokerInfo::connectionId [protected]

6.102.3.6 bool activemq::commands::BrokerInfo::duplexConnection [protected]

6.102.3.7 bool activemq::commands::BrokerInfo::faultTolerantConfiguration [protected]

6.102.3.8 const unsigned char activemq::commands::BrokerInfo::ID_BROKERINFO = 2 [static]

6.102.3.9 bool activemq::commands::BrokerInfo::masterBroker [protected]

6.102.3.10 bool activemq::commands::BrokerInfo::networkConnection [protected]

6.102.3.11 std::string activemq::commands::BrokerInfo::networkProperties [protected]

6.102.3.12 std::vector< decaf::lang::Pointer<BrokerInfo> > activemq::commands::BrokerInfo::peerBrokerInfos [protected]

6.102.3.13 bool activemq::commands::BrokerInfo::slaveBroker [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**BrokerInfo.h**

6.103 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshalle Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 615).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual ~**BrokerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.103.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 615). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.103.2 Constructor & Destructor Documentation

6.103.2.1 activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]

6.103.2.2 virtual activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::~BrokerInfoMarshall () [inline, virtual]

6.103.3 Member Function Documentation

6.103.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.103.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.103.3.3 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.103.3.4 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.103.3.5 virtual int activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.103.3.6 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.103.3.7 virtual void activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**BrokerInfoMarshaller.h**

6.104 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshalle Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 619).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual ~**BrokerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.104.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 619). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.104.2 Constructor & Destructor Documentation

6.104.2.1 activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]

6.104.2.2 virtual activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::~BrokerInfoMarshall () [inline, virtual]

6.104.3 Member Function Documentation

6.104.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.104.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.104.3.3 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.104.3.4 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.104.3.5 virtual int activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.104.3.6 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.104.3.7 virtual void activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**BrokerInfoMarshaller.h**

6.105 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshalle Class Reference

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 623).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller:

Public Member Functions

- **BrokerInfoMarshaller** ()
- virtual ~**BrokerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.105.1 Detailed Description

Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 623). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.105.2 Constructor & Destructor Documentation

6.105.2.1 activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::BrokerInfoMarshaller () [inline]

6.105.2.2 virtual activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::~BrokerInfoMarshall () [inline, virtual]

6.105.3 Member Function Documentation

6.105.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.105.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.105.3.3 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.105.3.4 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.105.3.5 virtual int activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.105.3.6 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.105.3.7 virtual void activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**BrokerInfoMarshaller.h**

6.106 decaf::nio::Buffer Class Reference

A container for data of a specific primitive type.

`#include <src/main/decaf/nio/Buffer.h>`Inheritance diagram for decaf::nio::Buffer:

Public Member Functions

- **Buffer** (std::size_t capactiy)
- **Buffer** (const **Buffer** &other)
- virtual ~**Buffer** ()
- virtual std::size_t **capacity** () const
- virtual std::size_t **position** () const
- virtual **Buffer** & **position** (std::size_t newPosition) throw (lang::exceptions::IllegalArgumentException)

  Sets this buffer's position.

- virtual std::size_t **limit** () const
- virtual **Buffer** & **limit** (std::size_t newLimit) throw (lang::exceptions::IllegalArgumentException)

  Sets this buffer's limit.

- virtual **Buffer** & **mark** ()

  Sets this buffer's mark at its position.

- virtual **Buffer** & **reset** () throw (InvalidMarkException)

  Resets this buffer's position to the previously-marked position.

- virtual **Buffer** & **clear** ()

  Clears this buffer.

- virtual **Buffer** & **flip** ()

  Flips this buffer.

- virtual **Buffer** & **rewind** ()

  Rewinds this buffer.

- virtual std::size_t **remaining** () const

  Returns the number of elements between the current position and the limit.

- virtual bool **hasRemaining** () const

  Tells whether there are any elements between the current position and the limit.

- virtual bool **isReadOnly** () const =0

  Tells whether or not this buffer is read-only.

Protected Attributes

- std::size_t **_position**
- std::size_t **_capacity**
- std::size_t **_limit**
- std::size_t **_mark**
- bool **_markSet**

6.106.1 Detailed Description

A container for data of a specific primitive type. A buffer is a linear, finite sequence of elements of a specific primitive type. Aside from its content, the essential properties of a buffer are its capacity, limit, and position:

A buffer's capacity is the number of elements it contains. The capacity of a buffer is never negative and never changes.

A buffer's limit is the index of the first element that should not be read or written. A buffer's limit is never negative and is never greater than its capacity.

A buffer's position is the index of the next element to be read or written. A buffer's position is never negative and is never greater than its limit.

There is one subclass of this class for each non-boolean primitive type.

Transferring data: Each subclass of this class defines two categories of get and put operations: * Relative operations read or write one or more elements starting at the current position and then increment the position by the number of elements transferred. If the requested transfer exceeds the limit then a relative get operation throws a **BufferUnderflowException** (p. 661) and a relative put operation throws a **BufferOverflowException** (p. 658); in either case, no data is transferred. * Absolute operations take an explicit element index and do not affect the position. Absolute get and put operations throw an IndexOutOfBoundsException if the index argument exceeds the limit.

Data may also, of course, be transferred in to or out of a buffer by the I/O operations of an appropriate channel, which are always relative to the current position.

Marking and resetting:

A buffer's mark is the index to which its position will be reset when the reset method is invoked. The mark is not always defined, but when it is defined it is never negative and is never greater than the position. If the mark is defined then it is discarded when the position or the limit is adjusted to a value smaller than the mark. If the mark is not defined then invoking the reset method causes an **InvalidMarkException** (p. 1470) to be thrown.

Invariants:

The following invariant holds for the mark, position, limit, and capacity values: 0 <= mark <= position <= limit <= capacity

A newly-created buffer always has a position of zero and a mark that is undefined. The initial limit may be zero, or it may be some other value that depends upon the type of the buffer and the manner in which it is constructed. The initial content of a buffer is, in general, undefined.

Clearing, flipping, and rewinding:

In addition to methods for accessing the position, limit, and capacity values and for marking and resetting, this class also defines the following operations upon buffers:

clear() (p. 629) makes a buffer ready for a new sequence of channel-read or relative put operations: It sets the limit to the capacity and the position to zero.

flip() (p. 630) makes a buffer ready for a new sequence of channel-write or relative get operations: It sets the limit to the current position and then sets the position to zero.

rewind() (p. 632) makes a buffer ready for re-reading the data that it already contains: It leaves the limit unchanged and sets the position to zero.

Read-only buffers:

Every buffer is readable, but not every buffer is writable. The mutation methods of each buffer class are specified as optional operations that will throw a **ReadOnlyBufferException** (p. 2167) when invoked upon a read-only buffer. A read-only buffer does not allow its content to be changed, but its mark, position, and limit values are mutable. Whether or not a buffer is read-only may be determined by invoking its isReadOnly method.

Thread safety:

Buffers are not safe for use by multiple concurrent threads. If a buffer is to be used by more than one thread then access to the buffer should be controlled by appropriate synchronization.

Invocation chaining:

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained; for example, the sequence of statements

b.flip(); b.position(23); b.limit(42);

can be replaced by the single, more compact statement b.flip().position(23).limit(42);

6.106.2 Constructor & Destructor Documentation

6.106.2.1 decaf::nio::Buffer::Buffer (std::size_t *capactiy*)

6.106.2.2 decaf::nio::Buffer::Buffer (const Buffer & *other*)

6.106.2.3 virtual decaf::nio::Buffer::~Buffer () [inline, virtual]

6.106.3 Member Function Documentation

6.106.3.1 virtual std::size_t decaf::nio::Buffer::capacity () const [inline, virtual]

Returns:

this buffer's capacity.

6.106.3.2 virtual Buffer& decaf::nio::Buffer::clear () [virtual]

Clears this buffer. The position is set to zero, the limit is set to the capacity, and the mark is discarded.

Invoke this method before using a sequence of channel-read or put operations to fill this buffer. For example:

buf.clear(); // Prepare buffer for reading in.read(buf); // Read data

This method does not actually erase the data in the buffer, but it is named as if it did because it will most often be used in situations in which that might as well be the case.

Returns:

a reference to this buffer.

6.106.3.3 virtual Buffer& decaf::nio::Buffer::flip () [virtual]

Flips this buffer. The limit is set to the current position and then the position is set to zero. If the mark is defined then it is discarded.

After a sequence of channel-read or put operations, invoke this method to prepare for a sequence of channel-write or relative get operations. For example:

buf.put(magic); // Prepend header in.read(buf); // Read data into rest of buffer buf.flip(); // Flip buffer out.write(buf); // Write header + data to channel

This method is often used in conjunction with the compact method when transferring data from one place to another.

Returns:

a reference to this buffer.

6.106.3.4 virtual bool decaf::nio::Buffer::hasRemaining () const [inline, virtual]

Tells whether there are any elements between the current position and the limit.

Returns:

true if, and only if, there is at least one element remaining in this buffer

6.106.3.5 virtual bool decaf::nio::Buffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 708), **decaf::internal::nio::CharArrayBuffer** (p. 814), **decaf::internal::nio::DoubleArrayBuffer** (p. 1247), **decaf::internal::nio::FloatArrayBuffer** (p. 1344), **decaf::internal::nio::IntArrayBuffer** (p. 1415), **decaf::internal::nio::LongArrayBuffer** (p. 1671), **decaf::internal::nio::ShortArrayBuffer** (p. 2335), and **decaf::nio::ByteBuffer** (p. 749).

6.106.3.6 virtual Buffer& decaf::nio::Buffer::limit (std::size_t *newLimit*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's limit. If the position is larger than the new limit then it is set to the new limit. If the mark is defined and larger than the new limit then it is discarded.

Parameters:

newLimit - The new limit value; must be no larger than this buffer's capacity

Returns:

A reference to This buffer

Exceptions:

IllegalArgumentException if preconditions on the new pos don't hold.

6.106.3.7 virtual std::size_t decaf::nio::Buffer::limit () const [inline, virtual]

Returns:

this buffers Limit

6.106.3.8 virtual Buffer& decaf::nio::Buffer::mark () [virtual]

Sets this buffer's mark at its position.

Returns:

a reference to this buffer.

6.106.3.9 virtual Buffer& decaf::nio::Buffer::position (std::size_t *newPosition*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Sets this buffer's position. If the mark is defined and larger than the new position then it is discarded.

Parameters:

newPosition - the new postion in the buffer to set.

Returns:

A reference to This buffer

Exceptions:

IllegalArgumentException if preconditions on the new pos don't hold.

6.106.3.10 virtual std::size_t decaf::nio::Buffer::position () const [inline, virtual]

Returns:

the current position in the buffer

6.106.3.11 virtual std::size_t decaf::nio::Buffer::remaining () const [inline, virtual]

Returns the number of elements between the current position and the limit.

Returns:

The number of elements remaining in this buffer

6.106.3.12 virtual Buffer& decaf::nio::Buffer::reset () throw (InvalidMarkException) [virtual]

Resets this buffer's position to the previously-marked position.

Returns:

a reference to this buffer.

Exceptions:

InvalidMarkException *(*p. *1470)* - If the mark has not been set

6.106.3.13 virtual Buffer& decaf::nio::Buffer::rewind () [virtual]

Rewinds this buffer. The position is set to zero and the mark is discarded.

Invoke this method before a sequence of channel-write or get operations, assuming that the limit has already been set appropriately. For example:

out.write(buf); // Write remaining data buf.rewind(); // Rewind buffer buf.get(array); // Copy data into array

Returns:

a reference to this buffer.

6.106.4 Field Documentation

6.106.4.1 std::size_t decaf::nio::Buffer::_capacity [protected]

6.106.4.2 std::size_t decaf::nio::Buffer::_limit [protected]

6.106.4.3 std::size_t decaf::nio::Buffer::_mark [protected]

6.106.4.4 bool decaf::nio::Buffer::_markSet [protected]

6.106.4.5 std::size_t decaf::nio::Buffer::_position [mutable, protected]

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**Buffer.h**

6.107 decaf::io::BufferedInputStream Class Reference

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of **io** (p. 100) operations on the input stream.

`#include <src/main/decaf/io/BufferedInputStream.h>`Inheritance diagram for decaf::io::BufferedInputStream:

Public Member Functions

- **BufferedInputStream** (**InputStream** *stream, bool **own**=false)

  Constructor.

- **BufferedInputStream** (**InputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)

  Constructor.

- virtual ~**BufferedInputStream** ()
- virtual std::size_t **available** () const throw (IOException)

  Indicates the number of bytes available.

- virtual void **close** () throw (IOException)

  *Close this **BufferedInputStream** (p. 633).*

- virtual unsigned char **read** () throw (IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

  Determines if this input stream supports the mark and reset methods.

6.107.1 Detailed Description

A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of **io** (p. 100) operations on the input stream.

6.107.2 Constructor & Destructor Documentation

6.107.2.1 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * *stream*, bool *own* = `false`)

Constructor.

Parameters:

stream The target input stream.

own indicates if we own the stream object, defaults to false

6.107.2.2 decaf::io::BufferedInputStream::BufferedInputStream (InputStream * *stream*, std::size_t *bufferSize*, bool *own* = `false`) throw (lang::exceptions::IllegalArgumentException)

Constructor.

Parameters:

stream the target input stream

bufferSize the size for the **internal** (p. 95) buffer.

own indicates if we own the stream object, defaults to false.

Exceptions:

IllegalArgumentException is the size is zero.

6.107.2.3 virtual decaf::io::BufferedInputStream::~BufferedInputStream () `[virtual]`

6.107.3 Member Function Documentation

6.107.3.1 virtual std::size_t decaf::io::BufferedInputStream::available () const throw (IOException) `[inline, virtual]`

Indicates the number of bytes available.

Returns:

the sum of the amount of data available in the buffer and the data available on the target input stream.

Reimplemented from **decaf::io::FilterInputStream** (p. 1317).

6.107.3.2 virtual void decaf::io::BufferedInputStream::close () throw (IOException) [virtual]

Close this **BufferedInputStream** (p. 633). This implementation closes the target stream and releases any resources associated with it.

Exceptions:

IOException *(p. 1477)* If an error occurs attempting to close this stream.

Reimplemented from **decaf::io::FilterInputStream** (p. 1317).

6.107.3.3 virtual void decaf::io::BufferedInputStream::mark (int readLimit) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit max bytes read before marked position is invalid.

Reimplemented from **decaf::io::FilterInputStream** (p. 1318).

6.107.3.4 virtual bool decaf::io::BufferedInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Reimplemented from **decaf::io::FilterInputStream** (p. 1318).

6.107.3.5 virtual int decaf::io::BufferedInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

NullPointerException if buffer is NULL

Reimplemented from **decaf::io::FilterInputStream** (p. 1319).

6.107.3.6 virtual unsigned char decaf::io::BufferedInputStream::read () throw (IOException) [virtual]

Reads a single byte from the buffer. Blocks until the data is available.

Returns:

The next byte.

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1319).

6.107.3.7 virtual void decaf::io::BufferedInputStream::reset () throw (IOException) [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: ∗ If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1477) might be thrown. ∗ If such an **IOException** (p. 1477) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: ∗ The call to reset may throw an **IOException** (p. 1477). ∗ If an **IOException** (p. 1477) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException *(*p. *1477)*

Reimplemented from **decaf::io::FilterInputStream** (p. 1320).

6.107.3.8 virtual std::size_t decaf::io::BufferedInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from

any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1406) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total butes skipped

Exceptions:

IOException *(*p. ***1477)*** if an error occurs

Reimplemented from **decaf::io::FilterInputStream** (p. 1320).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedInputStream.h**

6.108 decaf::io::BufferedOutputStream Class Reference

Wrapper around another output stream that buffers output before writing to the target output stream.

`#include <src/main/decaf/io/BufferedOutputStream.h>`Inheritance diagram for decaf::io::BufferedOutputStream:

Public Member Functions

- **BufferedOutputStream** (**OutputStream** *stream, bool **own**=false)

  Constructor.

- **BufferedOutputStream** (**OutputStream** *stream, std::size_t bufferSize, bool **own**=false) throw (lang::exceptions::IllegalArgumentException)

  Constructor.

- virtual ~**BufferedOutputStream** ()
- virtual void **write** (unsigned char c) throw (IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (IOException)

  Invokes flush on the target output stream.

- void **close** () throw (lang::Exception)

  Invokes close on the target output stream.

6.108.1 Detailed Description

Wrapper around another output stream that buffers output before writing to the target output stream.

6.108.2 Constructor & Destructor Documentation

6.108.2.1 decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * *stream*, bool *own* = `false`)

Constructor.

Parameters:

stream The target output stream.

own Indicates if this class owns the stream pointer.

6.108.2.2 decaf::io::BufferedOutputStream::BufferedOutputStream (OutputStream * *stream*, std::size_t *bufferSize*, bool *own* = `false`) throw (lang::exceptions::IllegalArgumentException)

Constructor.

Parameters:

stream The target output stream.

bufferSize The size for the **internal** (p. 95) buffer.

own Indicates if this class owns the stream pointer.

6.108.2.3 virtual decaf::io::BufferedOutputStream::~BufferedOutputStream () `[virtual]`

6.108.3 Member Function Documentation

6.108.3.1 void decaf::io::BufferedOutputStream::close () throw (lang::Exception) `[virtual]`

Invokes close on the target output stream.

Exceptions:

CMSException thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1324).

6.108.3.2 virtual void decaf::io::BufferedOutputStream::flush () throw (IOException) `[virtual]`

Invokes flush on the target output stream.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1324).

6.108.3.3 virtual void decaf::io::BufferedOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) `[virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException *(* p. *1477)* thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1326).

6.108.3.4 virtual void decaf::io::BufferedOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException *(* p. *1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1326).

6.108.3.5 virtual void decaf::io::BufferedOutputStream::write (unsigned char *c*) throw (IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException *(* p. *1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1327).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**BufferedOutputStream.h**

6.109 decaf::net::BufferedSocket Class Reference

Buffered **Socket** (p. 2371) class that wraps a `Socket (p. 2371)` derived object and provides Buffered input and Output Streams to improce the efficiency of the reads and writes.

`#include <src/main/decaf/net/BufferedSocket.h>`Inheritance diagram for decaf::net::BufferedSocket:

Public Member Functions

- **BufferedSocket** (**Socket** *socket, int inputBufferSize=1000, int outputBufferSize=1000, bool own=true)

  Constructs a new Buffered socket object.

- virtual ~**BufferedSocket** ()
- virtual void **connect** (const char *host, int port) throw (SocketException)

  Connects to the specified destination.

- virtual void **close** () throw (lang::Exception)

  Closes this object and deallocates the appropriate resources.

- virtual bool **isConnected** () const

  Indicates whether or not this socket is connected to a destination.

- virtual **io::InputStream** * **getInputStream** ()

  Gets the InputStream for this socket.

- virtual **io::OutputStream** * **getOutputStream** ()

  Gets the OutputStream for this socket.

- virtual int **getSoLinger** () const throw (SocketException)

  Gets the linger time.

- virtual void **setSoLinger** (int linger) throw (SocketException)

  Sets the linger time.

- virtual bool **getKeepAlive** () const throw (SocketException)

  Gets the keep alive flag.

- virtual void **setKeepAlive** (bool keepAlive) throw (SocketException)

  Enables/disables the keep alive flag.

- virtual int **getReceiveBufferSize** () const throw (SocketException)

  Gets the receive buffer size.

- virtual void **setReceiveBufferSize** (int size) throw (SocketException)

  Sets the recieve buffer size.

- virtual bool **getReuseAddress** () const throw (SocketException)
  Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
  Sets the reuse address flag.
- virtual int **getSendBufferSize** () const throw (SocketException)
  Gets the send buffer size.
- virtual void **setSendBufferSize** (int size) throw (SocketException)
  Sets the send buffer size.
- virtual int **getSoTimeout** () const throw (SocketException)
  Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout) throw (SocketException)
  Sets the timeout for socket operations.

6.109.1 Detailed Description

Buffered **Socket** (p. 2371) class that wraps a `Socket (p. 2371)` derived object and provides Buffered input and Output Streams to improce the efficiency of the reads and writes.

6.109.2 Constructor & Destructor Documentation

6.109.2.1 decaf::net::BufferedSocket::BufferedSocket (Socket * *socket*, int *inputBufferSize* = 1000, int *outputBufferSize* = 1000, bool *own* = true)

Constructs a new Buffered socket object.

Parameters:

socket the socket to buffer
inputBufferSize size of the input buffer
outputBufferSize size of the output buffer
own does this object own the passed socket

6.109.2.2 virtual decaf::net::BufferedSocket::~BufferedSocket () [virtual]

6.109.3 Member Function Documentation

6.109.3.1 virtual void decaf::net::BufferedSocket::close () throw (lang::Exception) [virtual]

Closes this object and deallocates the appropriate resources.

Exceptions:

CMSException

Implements **decaf::io::Closeable** (p. 840).

6.109.3.2 virtual void decaf::net::BufferedSocket::connect (const char * *host*, int *port*) throw (SocketException) [virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

IOException Thrown if a failure occurred in the connect.

Implements **decaf::net::Socket** (p. 2372).

6.109.3.3 virtual io::InputStream* decaf::net::BufferedSocket::getInputStream () [inline, virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2373).

6.109.3.4 virtual bool decaf::net::BufferedSocket::getKeepAlive () const throw (SocketException) [inline, virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException *(p. 2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2373).

References decaf::net::Socket::getKeepAlive().

6.109.3.5 virtual io::OutputStream* decaf::net::BufferedSocket::getOutputStream () [inline, virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2373).

6.109.3.6 virtual int decaf::net::BufferedSocket::getReceiveBufferSize () const throw (SocketException) [inline, virtual]

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException *(* p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2373).

References decaf::net::Socket::getReceiveBufferSize().

6.109.3.7 virtual bool decaf::net::BufferedSocket::getReuseAddress () const throw (SocketException) [inline, virtual]

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException *(* p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

References decaf::net::Socket::getReuseAddress().

6.109.3.8 virtual int decaf::net::BufferedSocket::getSendBufferSize () const throw (SocketException) [inline, virtual]

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException *(* p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

References decaf::net::Socket::getSendBufferSize().

6.109.3.9 virtual int decaf::net::BufferedSocket::getSoLinger () const throw (SocketException) [inline, virtual]

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

References decaf::net::Socket::getSoLinger().

6.109.3.10 virtual int decaf::net::BufferedSocket::getSoTimeout () const throw (SocketException) [inline, virtual]

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(*p. *2379)* Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2375).

References decaf::net::Socket::getSoTimeout().

6.109.3.11 virtual bool decaf::net::BufferedSocket::isConnected () const [inline, virtual]

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implements **decaf::net::Socket** (p. 2375).

References decaf::net::Socket::isConnected().

6.109.3.12 virtual void decaf::net::BufferedSocket::setKeepAlive (bool *keepAlive*) throw (SocketException) [inline, virtual]

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2375).

References decaf::net::Socket::setKeepAlive().

6.109.3.13 virtual void decaf::net::BufferedSocket::setReceiveBufferSize (int *size*) throw (SocketException) [inline, virtual]

Sets the recieve buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2375).

References decaf::net::Socket::setReceiveBufferSize().

6.109.3.14 virtual void decaf::net::BufferedSocket::setReuseAddress (bool *reuse*) throw (SocketException) [inline, virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

References decaf::net::Socket::setReuseAddress().

6.109.3.15 virtual void decaf::net::BufferedSocket::setSendBufferSize (int *size*) throw (SocketException) [inline, virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

References decaf::net::Socket::setSendBufferSize().

6.109.3.16 virtual void decaf::net::BufferedSocket::setSoLinger (int *linger*) throw (SocketException) [inline, virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException *(* p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

References decaf::net::Socket::setSoLinger().

6.109.3.17 virtual void decaf::net::BufferedSocket::setSoTimeout (int *timeout*) throw (SocketException) [inline, virtual]

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(* p. *2379)* Thrown if unable to set the information.

Implements **decaf::net::Socket** (p. 2377).

References decaf::net::Socket::setSoTimeout().

The documentation for this class was generated from the following file:

- src/main/decaf/net/**BufferedSocket.h**

6.110 decaf::internal::nio::BufferFactory Class Reference

Factory class used by static methods in the **decaf::nio** (p. 106) package to create the various default version of the NIO interfaces.

`#include <src/main/decaf/internal/nio/BufferFactory.h>`

Public Member Functions

- virtual ~**BufferFactory** ()

Static Public Member Functions

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (std::size_t capacity)

  Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new ByteBuffer.

- static **decaf::nio::ByteBuffer** * **createByteBuffer** (std::vector< unsigned char > &buffer)

  Wraps the passed STL Byte Vector in a ByteBuffer.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (std::size_t capacity)

  Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (char *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new CharBuffer.

- static **decaf::nio::CharBuffer** * **createCharBuffer** (std::vector< char > &buffer)

  Wraps the passed STL Byte Vector in a CharBuffer.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (std::size_t capacity)

  Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (double *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new DoubleBuffer.

- static **decaf::nio::DoubleBuffer** * **createDoubleBuffer** (std::vector< double > &buffer)

  Wraps the passed STL Double Vector in a DoubleBuffer.

- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (std::size_t capacity)

  Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (float *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new FloatBuffer.

- static **decaf::nio::FloatBuffer** * **createFloatBuffer** (std::vector< float > &buffer)

  Wraps the passed STL Float Vector in a FloatBuffer.

- static **decaf::nio::LongBuffer** * **createLongBuffer** (std::size_t capacity)

  Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::LongBuffer** * **createLongBuffer** (long long *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new LongBuffer.

- static **decaf::nio::LongBuffer** * **createLongBuffer** (std::vector< long long > &buffer)

  Wraps the passed STL Long Vector in a LongBuffer.

- static **decaf::nio::IntBuffer** * **createIntBuffer** (std::size_t capacity)

  Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::IntBuffer** * **createIntBuffer** (int *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new IntBuffer.

- static **decaf::nio::IntBuffer** * **createIntBuffer** (std::vector< int > &buffer)

  Wraps the passed STL int Vector in a IntBuffer.

- static **decaf::nio::ShortBuffer** * **createShortBuffer** (std::size_t capacity)

  Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **decaf::nio::ShortBuffer** * **createShortBuffer** (short *buffer, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  Wraps the passed buffer with a new ShortBuffer.

- static **decaf::nio::ShortBuffer** * **createShortBuffer** (std::vector< short > &buffer)

  Wraps the passed STL Short Vector in a ShortBuffer.

6.110.1 Detailed Description

Factory class used by static methods in the **decaf::nio** (p. 106) package to create the various default version of the NIO interfaces.

6.110.2 Constructor & Destructor Documentation

6.110.2.1 virtual decaf::internal::nio::BufferFactory::~BufferFactory () [inline, virtual]

6.110.3 Member Function Documentation

6.110.3.1 static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a ByteBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new ByteBuffer that is backed by buffer, caller owns.

6.110.3.2 static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new ByteBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new ByteBuffer that is backed by buffer, caller owns.

6.110.3.3 static decaf::nio::ByteBuffer* decaf::internal::nio::BufferFactory::createByteBuffer (std::size_t *capacity*) [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated ByteBuffer which the caller owns.

6.110.3.4 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::vector< char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a CharBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new CharBuffer that is backed by buffer, caller owns.

6.110.3.5 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new CharBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new CharBuffer that is backed by buffer, caller owns.

6.110.3.6 static decaf::nio::CharBuffer* decaf::internal::nio::BufferFactory::createCharBuffer (std::size_t *capacity*) [static]

Allocates a new char buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated CharBuffer which the caller owns.

6.110.3.7 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::vector< double > & *buffer*) [static]

Wraps the passed STL Double Vector in a DoubleBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.8 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (double * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new DoubleBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.9 static decaf::nio::DoubleBuffer* decaf::internal::nio::BufferFactory::createDoubleBuffer (std::size_t *capacity*) [static]

Allocates a new double buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.110.3.10 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::vector< float > & *buffer*) [static]

Wraps the passed STL Float Vector in a FloatBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.11 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new FloatBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.12 static decaf::nio::FloatBuffer* decaf::internal::nio::BufferFactory::createFloatBuffer (std::size_t *capacity*) [static]

Allocates a new float buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.110.3.13 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::vector< int > & *buffer*) [static]

Wraps the passed STL int Vector in a IntBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.14 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (int * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new IntBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.15 static decaf::nio::IntBuffer* decaf::internal::nio::BufferFactory::createIntBuffer (std::size_t *capacity*) [static]

Allocates a new int buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.110.3.16 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::vector< long long > & *buffer*) [static]

Wraps the passed STL Long Vector in a LongBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.17 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new LongBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.18 static decaf::nio::LongBuffer* decaf::internal::nio::BufferFactory::createLongBuffer (std::size_t *capacity*) [static]

Allocates a new long long buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

6.110.3.19 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::vector< short > & *buffer*) [static]

Wraps the passed STL Short Vector in a ShortBuffer. The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.20 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (short * *buffer*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new ShortBuffer. The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new DoubleBuffer that is backed by buffer, caller owns.

6.110.3.21 static decaf::nio::ShortBuffer* decaf::internal::nio::BufferFactory::createShortBuffer (std::size_t *capacity*) [static]

Allocates a new short buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated DoubleBuffer which the caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**BufferFactory.h**

6.111 decaf::nio::BufferOverflowException Class Reference

`#include <src/main/decaf/nio/BufferOverflowException.h>`Inheritance diagram for decaf::nio::BufferOverflowException:

Public Member Functions

- **BufferOverflowException** () throw ()
  Default Constructor.
- **BufferOverflowException** (const **lang::Exception** &ex) throw ()
  Copy Constructor.
- **BufferOverflowException** (const **BufferOverflowException** &ex) throw ()
  Copy Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **BufferOverflowException** (const std::exception ***cause**) throw ()
  Constructor.
- **BufferOverflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor.
- virtual **BufferOverflowException** * **clone** () const
  Clones this exception.
- virtual ~**BufferOverflowException** () throw ()

6.111.1 Constructor & Destructor Documentation

6.111.1.1 decaf::nio::BufferOverflowException::BufferOverflowException () throw () [inline]

Default Constructor.

6.111.1.2 decaf::nio::BufferOverflowException::BufferOverflowException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.111.1.3 decaf::nio::BufferOverflowException::BufferOverflowException (const BufferOverflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.111.1.4 decaf::nio::BufferOverflowException::BufferOverflowException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.111.1.5 decaf::nio::BufferOverflowException::BufferOverflowException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.111.1.6 decaf::nio::BufferOverflowException::BufferOverflowException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.111.1.7 virtual decaf::nio::BufferOverflowException::~BufferOverflowException () throw () [inline, virtual]

6.111.2 Member Function Documentation

6.111.2.1 virtual BufferOverflowException* decaf::nio::BufferOverflowException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferOverflowException.h**

6.112 decaf::nio::BufferUnderflowException Class Reference

`#include <src/main/decaf/nio/BufferUnderflowException.h>`Inheritance diagram for decaf::nio::BufferUnderflowException:

Public Member Functions

- **BufferUnderflowException** () throw ()

  Default Constructor.

- **BufferUnderflowException** (const **lang::Exception** &ex) throw ()

  Copy Constructor.

- **BufferUnderflowException** (const **BufferUnderflowException** &ex) throw ()

  Copy Constructor.

- **BufferUnderflowException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **BufferUnderflowException** (const std::exception ***cause**) throw ()

  Constructor.

- **BufferUnderflowException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor.

- virtual **BufferUnderflowException** * **clone** () const

  Clones this exception.

- virtual ~**BufferUnderflowException** () throw ()

6.112.1 Constructor & Destructor Documentation

6.112.1.1 decaf::nio::BufferUnderflowException::BufferUnderflowException () throw () `[inline]`

Default Constructor.

6.112.1.2 decaf::nio::BufferUnderflowException::BufferUnderflowException (const lang::Exception & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex the exception to copy

6.112.1.3 decaf::nio::BufferUnderflowException::BufferUnderflowException (const BufferUnderflowException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.112.1.4 decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.112.1.5 decaf::nio::BufferUnderflowException::BufferUnderflowException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.112.1.6 decaf::nio::BufferUnderflowException::BufferUnderflowException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.112.1.7 virtual decaf::nio::BufferUnderflowException::~BufferUnderflowException () throw () [inline, virtual]

6.112.2 Member Function Documentation

6.112.2.1 virtual BufferUnderflowException* decaf::nio::BufferUnderflowException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**BufferUnderflowException.h**

6.113 decaf::lang::Byte Class Reference

`#include <src/main/decaf/lang/Byte.h>`Inheritance diagram for decaf::lang::Byte:

Public Member Functions

- **Byte** (unsigned char value)
- **Byte** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Byte** ()
- virtual int **compareTo** (const **Byte** &c) const

  *Compares this **Byte** (p. 664) instance with another.*

- virtual bool **operator==** (const **Byte** &c) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Byte** &c) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const unsigned char &c) const

  *Compares this **Byte** (p. 664) instance with a char type.*

- virtual bool **operator==** (const unsigned char &c) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const unsigned char &c) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- bool **equals** (const **Byte** &c) const
- bool **equals** (const unsigned char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (unsigned char value)
- static **Byte decode** (const std::string &value) throw (exceptions::NumberFormatException)

  *Decodes a String into a **Byte** (p. 664).*

- static unsigned char **parseByte** (const std::string &s, int radix) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed unsigned char in the radix specified by the second argument.

- static unsigned char **parseByte** (const std::string &s) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed decimal unsigned char.

- static **Byte valueOf** (unsigned char value)

  *Returns a **Character** (p. 801) instance representing the specified char value.*

- static **Byte valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a **Byte** (p. 664) object holding the value given by the specified std::string.*

- static **Byte valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

  *Returns a **Byte** (p. 664) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const unsigned char **MIN_VALUE** = 0x7F

  The minimum value that a unsigned char can take on.

- static const unsigned char **MAX_VALUE** = 0x80

  The maximum value that a unsigned char can take on.

- static const int **SIZE** = 8

  The size of the primitive charactor in bits.

6.113.1 Constructor & Destructor Documentation

6.113.1.1 decaf::lang::Byte::Byte (unsigned char *value*)

Parameters:

value - the primitive value to wrap

6.113.1.2 decaf::lang::Byte::Byte (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the string to convert to an unsigned char

Exceptions:

NumberFormatException

6.113.1.3 virtual decaf::lang::Byte::~Byte () [inline, virtual]

6.113.2 Member Function Documentation

6.113.2.1 virtual unsigned char decaf::lang::Byte::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.113.2.2 virtual int decaf::lang::Byte::compareTo (const unsigned char & *c*) const [inline, virtual]

Compares this **Byte** (p. 664) instance with a char type.

Parameters:

c - the char instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 899).

6.113.2.3 virtual int decaf::lang::Byte::compareTo (const Byte & *c*) const [inline, virtual]

Compares this **Byte** (p. 664) instance with another.

Parameters:

c - the **Byte** (p. 664) instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

6.113.2.4 static Byte decaf::lang::Byte::decode (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Byte** (p. 664). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Byte::parseByte** (p. 670) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Byte** (p. 664) object containing the decoded value

Exceptions:

NumberFomatException if the string is not formatted correctly.

6.113.2.5 virtual double decaf::lang::Byte::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.113.2.6 bool decaf::lang::Byte::equals (const unsigned char & *c*) const [inline, virtual]

Returns:

true if the two Bytes have the same value.

Implements **decaf::lang::Comparable< unsigned char >** (p. 900).

6.113.2.7 bool decaf::lang::Byte::equals (const Byte & *c*) const [inline]

Returns:

true if the two **Byte** (p. 664) Objects have the same value.

6.113.2.8 virtual float decaf::lang::Byte::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.113.2.9 virtual int decaf::lang::Byte::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.113.2.10 virtual long long decaf::lang::Byte::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.113.2.11 virtual bool decaf::lang::Byte::operator< (const unsigned char & *c*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 900).

6.113.2.12 virtual bool decaf::lang::Byte::operator< (const Byte & *c*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.113.2.13 virtual bool decaf::lang::Byte::operator== (const unsigned char & *c*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< unsigned char >** (p. 901).

6.113.2.14 virtual bool decaf::lang::Byte::operator== (const Byte & *c*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.113.2.15 static unsigned char decaf::lang::Byte::parseByte (const std::string & *s*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal unsigned char. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting unsigned char value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseByte(const std::string, int) method.

Parameters:

s - String to convert to a unsigned char

Returns:

the converted unsigned char value

Exceptions:

NumberFormatException if the string is not a unsigned char.

6.113.2.16 static unsigned char decaf::lang::Byte::parseByte (const std::string & *s*, int *radix*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed unsigned char in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 804) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type NumberFormatException is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 808) or larger than **Character::MAX_RADIX** (p. 807). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type unsigned char.

Parameters:

s - the String containing the unsigned char to be parsed

radix - the radix to be used while parsing s

Returns:

the unsigned char represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable unsigned char.

6.113.2.17 virtual short decaf::lang::Byte::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.113.2.18 static std::string decaf::lang::Byte::toString (unsigned char *value*) [static]

Returns:

a string representing the primitive value as Base 10

6.113.2.19 std::string decaf::lang::Byte::toString () const

Returns:

this **Byte** (p. 664) Object as a String Representation

6.113.2.20 static Byte decaf::lang::Byte::valueOf (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Byte** (p. 664) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed unsigned char in the radix specified by the second argument, exactly as if the argument were given to the parseByte(std::string, int) method. The result is a **Byte** (p. 664) object that represents the unsigned char value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Byte** (p. 664) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid unsigned char.

6.113.2.21 static Byte decaf::lang::Byte::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Byte** (p. 664) object holding the value given by the specified std::string. The argument is interpreted as representing a signed decimal unsigned char, exactly as if the argument were given to the parseByte(std::string) method. The result is a **Byte** (p. 664) object that represents the unsigned char value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Byte** (p. 664) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal unsigned char.

6.113.2.22 static Byte decaf::lang::Byte::valueOf (unsigned char *value*) [inline, static]

Returns a **Character** (p. 801) instance representing the specified char value.

Parameters:

value - the primitive char to wrap.

Returns:

a new **Character** (p. 801) instance that wraps this value.

6.113.3 Field Documentation

6.113.3.1 const unsigned char decaf::lang::Byte::MAX_VALUE = 0x80 [static]

The maximum value that a unsigned char can take on.

6.113.3.2 const unsigned char decaf::lang::Byte::MIN_VALUE = 0x7F [static]

The minimum value that a unsigned char can take on.

6.113.3.3 const int decaf::lang::Byte::SIZE = 8 [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Byte.h**

6.114 decaf::internal::util::ByteArrayAdapter Class Reference

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

`#include <src/main/decaf/internal/util/ByteArrayAdapter.h>`Inheritance diagram for decaf::internal::util::ByteArrayAdapter:

Data Structures

- union **Array**

Public Member Functions

- **ByteArrayAdapter** (std::size_t capacity)

  Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

- **ByteArrayAdapter** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayAdapter** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- virtual ~**ByteArrayAdapter** ()

- virtual std::size_t **getCapacity** () const

  Gets the capacity of the underlying array.

- virtual std::size_t **getCharCapacity** () const

  Gets the capacity of the underlying array as if it contains chars.

- virtual std::size_t **getDoubleCapacity** () const

  Gets the capacity of the underlying array as if it contains doubles.

- virtual std::size_t **getFloatCapacity** () const

  Gets the capacity of the underlying array as if it contains doubles.

- virtual std::size_t **getLongCapacity** () const

  Gets the capacity of the underlying array as if it contains doubles.

- virtual std::size_t **getIntCapacity** () const

  Gets the capacity of the underlying array as if it contains ints.

- virtual std::size_t **getShortCapacity** () const

  Gets the capacity of the underlying array as if it contains shorts.

- virtual unsigned char * **getByteArray** ()

  Gets the pointer to the array we are wrapping.

- virtual char * **getCharArray** ()

  Gets the pointer to the array we are wrapping.

- virtual short * **getShortArray** ()

  Gets the pointer to the array we are wrapping.

- virtual int * **getIntArray** ()

  Gets the pointer to the array we are wrapping.

- virtual long long * **getLongArray** ()

  Gets the pointer to the array we are wrapping.

- virtual double * **getDoubleArray** ()

  Gets the pointer to the array we are wrapping.

- virtual float * **getFloatArray** ()

  Gets the pointer to the array we are wrapping.

- virtual void **read** (unsigned char *buffer, std::size_t offset, std::size_t length) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException)

  Reads from the Byte array starting at the specified offset and reading the specified length.

- virtual void **write** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException)

  *Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects **internal** (p. 95) array.*

- virtual void **resize** (std::size_t capacity) throw (lang::exceptions::InvalidStateException)

  Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved.

- virtual void **clear** ()

  Clear all data from that Array, setting the underlying bytes to zero.

- unsigned char & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Allows the **ByteArrayAdapter** (p. 673) to be indexed as a standard array.*

- const unsigned char & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual unsigned char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- virtual char **getChar** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads one byte at the given index and returns it.

- virtual double **getDouble** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given index and returns it.

- virtual double **getDoubleAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given byte index and returns it.

- virtual float **getFloat** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given index and returns it.

- virtual float **getFloatAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given byte index and returns it.

- virtual long long **getLong** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given index and returns it.

- virtual long long **getLongAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given byte index and returns it.

- virtual int **getInt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given index and returns it.

- virtual int **getIntAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given byte index and returns it.

- virtual short **getShort** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads two bytes at the given index and returns it.

- virtual short **getShortAt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads two bytes at the given byte index and returns it.

- virtual **ByteArrayAdapter** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes the given byte into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putDoubleAt** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putFloatAt** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putLongAt** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes eight bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putIntAt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes four bytes containing the given value, into this buffer at the given byte index.

- virtual **ByteArrayAdapter** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayAdapter** & **putShortAt** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException)

  Writes two bytes containing the given value, into this buffer at the given byte index.

6.114.1 Detailed Description

This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data. All the array types are mapped down to a byte array and methods are supplied for accessing the data in any of the primitive type forms.

Methods in this class that do not return a specific value return a reference to this object so that calls can be chained.

6.114.2 Constructor & Destructor Documentation

6.114.2.1 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (std::size_t *capacity*)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.114.2.2 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (unsigned char * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.3 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (char * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.4 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (double * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.5 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (float * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.6 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (long long * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.7 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (int * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.8 decaf::internal::util::ByteArrayAdapter::ByteArrayAdapter (short * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.114.2.9 virtual decaf::internal::util::ByteArrayAdapter::~ByteArrayAdapter () [virtual]

6.114.3 Member Function Documentation

6.114.3.1 virtual void decaf::internal::util::ByteArrayAdapter::clear () [virtual]

Clear all data from that Array, setting the underlying bytes to zero.

6.114.3.2 virtual unsigned char decaf::internal::util::ByteArrayAdapter::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.114.3.3 virtual unsigned char* decaf::internal::util::ByteArrayAdapter::getByteArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an unsigned char* pointer to the array this object wraps.

6.114.3.4 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCapacity () const [inline, virtual]

Gets the capacity of the underlying array.

Returns:

the size the array.

6.114.3.5 virtual char decaf::internal::util::ByteArrayAdapter::getChar (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Reads one byte at the given index and returns it.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

6.114.3.6 virtual char* decaf::internal::util::ByteArrayAdapter::getCharArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an char* pointer to the array this object wraps.

6.114.3.7 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getCharCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains chars.

Returns:

the size the array.

6.114.3.8 virtual double decaf::internal::util::ByteArrayAdapter::getDouble (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in otherwords when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.9 virtual double* decaf::internal::util::ByteArrayAdapter::getDoubleArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an double* pointer to the array this object wraps.

6.114.3.10 virtual double decaf::internal::util::ByteArrayAdapter::getDoubleAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.11 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getDoubleCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.114.3.12 virtual float decaf::internal::util::ByteArrayAdapter::getFloat (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.13 virtual float* decaf::internal::util::ByteArrayAdapter::getFloatArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an float* pointer to the array this object wraps.

6.114.3.14 virtual float decaf::internal::util::ByteArrayAdapter::getFloatAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.15 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getFloatCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.114.3.16 virtual int decaf::internal::util::ByteArrayAdapter::getInt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.17 virtual int* decaf::internal::util::ByteArrayAdapter::getIntArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an int* pointer to the array this object wraps.

6.114.3.18 virtual int decaf::internal::util::ByteArrayAdapter::getIntAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.19 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getIntCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains ints.

Returns:

the size the array.

6.114.3.20 virtual long long decaf::internal::util::ByteArrayAdapter::getLong (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.21 virtual long long* decaf::internal::util::ByteArrayAdapter::getLongArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an long long* pointer to the array this object wraps.

6.114.3.22 virtual long long decaf::internal::util::ByteArrayAdapter::getLongAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.23 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getLongCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains doubles.

Returns:

the size the array.

6.114.3.24 virtual short decaf::internal::util::ByteArrayAdapter::getShort (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads two bytes at the given index and returns it. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.25 virtual short* decaf::internal::util::ByteArrayAdapter::getShortArray () [inline, virtual]

Gets the pointer to the array we are wrapping. Changes to the data in this array are reflected by all **ByteArrayAdapter** (p. 673) objects that point to this array.

Returns:

an short* pointer to the array this object wraps.

6.114.3.26 virtual short decaf::internal::util::ByteArrayAdapter::getShortAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads two bytes at the given byte index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

6.114.3.27 virtual std::size_t decaf::internal::util::ByteArrayAdapter::getShortCapacity () const [inline, virtual]

Gets the capacity of the underlying array as if it contains shorts.

Returns:

the size the array.

6.114.3.28 const unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

6.114.3.29 unsigned char& decaf::internal::util::ByteArrayAdapter::operator[] (std::size_t *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Allows the **ByteArrayAdapter** (p. 673) to be indexed as a standard array. calling the non const version allows the user to change the value at index

Parameters:

index - the position in the array to access

Exceptions:

IndexOutOfBoundsException

6.114.3.30 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::put (std::size_t *index*, unsigned char *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes the given byte into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.31 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putChar (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes one byte containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.32 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDouble (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.33 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putDoubleAt (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.34 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.35 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putFloatAt (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.36 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putInt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.37 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putIntAt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes four bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.38 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLong (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.39 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putLongAt (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.40 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShort (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes two bytes containing the given value, into this buffer at the given index. The index is a relative to the size of the type to be read, in other words when accessing the element in the array index * sizeof(type) if the actual start index of the type to be read.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.41 virtual ByteArrayAdapter& decaf::internal::util::ByteArrayAdapter::putShortAt (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Writes two bytes containing the given value, into this buffer at the given byte index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

6.114.3.42 virtual void decaf::internal::util::ByteArrayAdapter::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) const throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferUnderflowException) [virtual]

Reads from the Byte array starting at the specified offset and reading the specified length. If the length is greater than the capacity of this underlying byte array then an BufferUnderflowException is thrown.

Parameters:

buffer - the buffer to read data from this array into.

offset - position in this array to start reading from.

length - the amount of data to read from this array.

Exceptions:

NullPointerException if buffer is null

BufferUnderflowException if there is not enought data to read because the offset or the length is greater than the size of this array.

6.114.3.43 virtual void decaf::internal::util::ByteArrayAdapter::resize (std::size_t *capacity*) throw (lang::exceptions::InvalidStateException) [virtual]

Resizes the underlying array to the new given capacity, preserving all the Data that was previously in the array, unless the resize is smaller than the current size in which case only the data that will fit into the new array is preserved. A **ByteArrayAdapter** (p. 673) can only be resized when it owns the underlying array, if it does not then it will throw an IllegalStateException.

Parameters:

capacity - the new capacity of the array.

Exceptions:

InvalidStateException if this object does not own the buffer.

6.114.3.44 virtual void decaf::internal::util::ByteArrayAdapter::write (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (decaf::lang::exceptions::NullPointerException, decaf::nio::BufferOverflowException) [virtual]

Writes from the Byte array given, starting at the specified offset and writing the specified amount of data into this objects **internal** (p. 95) array. . If the length is greater than the capacity of this underlying byte array then an BufferOverflowException is thrown.

Parameters:

buffer - the buffer to write get data written into this array.

offset - position in this array to start writing from.

length - the amount of data to write to this array.

Exceptions:

NullPointerException if buffer is null

BufferOverflowException if the amount of data to be written to this array or the offset given are larger than this array's capacity.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**ByteArrayAdapter.h**

6.115 decaf::internal::nio::ByteArrayBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

`#include <src/main/decaf/internal/nio/ByteArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::ByteArrayBuffer:

Public Member Functions

- **ByteArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **ByteArrayBuffer** (p. 694) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **ByteArrayBuffer** (unsigned char *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **ByteArrayBuffer** (p. 694) object that wraps the given array.*

- **ByteArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t length, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **ByteArrayBuffer** (const **ByteArrayBuffer** &other)

  *Create a **ByteArrayBuffer** (p. 694) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**ByteArrayBuffer** ()
- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual unsigned char * **array** () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException)

  Returns the byte array that backs this buffer.

- virtual std::size_t **arrayOffset** () const throw (decaf::nio::ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)

  Returns the offset within this buffer's backing array of the first element of the buffer.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible byte array.

- virtual **decaf::nio::CharBuffer** * **asCharBuffer** () const

  Creates a view of this byte buffer as a char buffer.

- virtual **decaf::nio::DoubleBuffer** * **asDoubleBuffer** () const

  Creates a view of this byte buffer as a double buffer.

- virtual **decaf::nio::FloatBuffer** * **asFloatBuffer** () const

Creates a view of this byte buffer as a float buffer.

- virtual **decaf::nio::IntBuffer** * **asIntBuffer** () const
  Creates a view of this byte buffer as a int buffer.
- virtual **decaf::nio::LongBuffer** * **asLongBuffer** () const
  Creates a view of this byte buffer as a long buffer.
- virtual **decaf::nio::ShortBuffer** * **asShortBuffer** () const
  Creates a view of this byte buffer as a short buffer.
- virtual **ByteArrayBuffer** * **asReadOnlyBuffer** () const
  Creates a new, read-only byte buffer that shares this buffer's content.
- virtual **ByteArrayBuffer** & **compact** () throw (decaf::nio::ReadOnlyBufferException)
  Compacts this buffer.
- virtual **ByteArrayBuffer** * **duplicate** ()
  Creates a new byte buffer that shares this buffer's content.
- virtual unsigned char **get** () const throw (decaf::nio::BufferUnderflowException)
  Relative get method.
- virtual unsigned char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
  Absolute get method.
- virtual char **getChar** () throw (decaf::nio::BufferUnderflowException)
  Reads the next byte at this buffer's current position, and then increments the position by one.
- virtual char **getChar** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
  Reads one byte at the given index and returns it.
- virtual double **getDouble** () throw (decaf::nio::BufferUnderflowException)
  Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.
- virtual double **getDouble** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
  Reads eight bytes at the given index and returns it.
- virtual float **getFloat** () throw (decaf::nio::BufferUnderflowException)
  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.
- virtual float **getFloat** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)
  Reads four bytes at the given index and returns it.
- virtual long long **getLong** () throw (decaf::nio::BufferUnderflowException)

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual long long **getLong** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given index and returns it.

- virtual int **getInt** () throw (decaf::nio::BufferUnderflowException)

  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual int **getInt** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given index and returns it.

- virtual short **getShort** () throw (decaf::nio::BufferUnderflowException)

  Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

- virtual short **getShort** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Reads two bytes at the given index and returns it.

- virtual **ByteArrayBuffer** & **put** (unsigned char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given byte into this buffer at the current position, and then increments the position.

- virtual **ByteArrayBuffer** & **put** (std::size_t index, unsigned char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given byte into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putChar** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

- virtual **ByteArrayBuffer** & **putChar** (std::size_t index, char value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putDouble** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteArrayBuffer** & **putDouble** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putFloat** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteArrayBuffer** & **putFloat** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putLong** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteArrayBuffer** & **putLong** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putInt** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteArrayBuffer** & **putInt** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** & **putShort** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteArrayBuffer** & **putShort** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteArrayBuffer** * **slice** () const

  Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.115.1 Detailed Description

This class defines six categories of operations upon byte buffers:. 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer;

4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 706) float getFloat(int index) void **putFloat(float f)** (p. 712) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the FloatBuffer class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.115.2 Constructor & Destructor Documentation

6.115.2.1 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **ByteArrayBuffer** (p. 694) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.115.2.2 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (unsigned char * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **ByteArrayBuffer** (p. 694) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.115.2.3 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *length*, bool *readOnly* = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **ByteArrayBuffer** (p. 694) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.115.2.4 decaf::internal::nio::ByteArrayBuffer::ByteArrayBuffer (const ByteArrayBuffer & *other*)

Create a **ByteArrayBuffer** (p. 694) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **ByteArrayBuffer** (p. 694) this one is to mirror.

6.115.2.5 virtual decaf::internal::nio::ByteArrayBuffer::~ByteArrayBuffer () [virtual]

6.115.3 Member Function Documentation

6.115.3.1 virtual unsigned char* decaf::internal::nio::ByteArrayBuffer::array () throw (decaf::nio::ReadOnlyBufferException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Returns the byte array that backs this buffer. Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The array that backs this buffer

Exceptions:

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 739).

6.115.3.2 virtual std::size_t decaf::internal::nio::ByteArrayBuffer::arrayOffset () const throw (decaf::nio::ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position p corresponds to array index p + **arrayOffset()** (p. 700).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset within this buffer's array of the first element of the buffer

Exceptions:

ReadOnlyBufferException - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implements **decaf::nio::ByteBuffer** (p. 740).

6.115.3.3 virtual decaf::nio::CharBuffer* decaf::internal::nio::ByteArrayBuffer::asCharBuffer () const [inline, virtual]

Creates a view of this byte buffer as a char buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new Char Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 740).

6.115.3.4 virtual decaf::nio::DoubleBuffer* decaf::internal::nio::ByteArrayBuffer::asDoubleBuffer () const [inline, virtual]

Creates a view of this byte buffer as a double buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new double Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 740).

6.115.3.5 virtual decaf::nio::FloatBuffer* decaf::internal::nio::ByteArrayBuffer::asFloatBuffer () const [inline, virtual]

Creates a view of this byte buffer as a float buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new float Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 741).

6.115.3.6 virtual decaf::nio::IntBuffer* decaf::internal::nio::ByteArrayBuffer::asIntBuffer () const [inline, virtual]

Creates a view of this byte buffer as a int buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new int Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 741).

6.115.3.7 virtual decaf::nio::LongBuffer* decaf::internal::nio::ByteArrayBuffer::asLongBuffer () const [inline, virtual]

Creates a view of this byte buffer as a long buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new long Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 741).

6.115.3.8 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only byte buffer which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 742).

6.115.3.9 virtual decaf::nio::ShortBuffer* decaf::internal::nio::ByteArrayBuffer::asShortBuffer () const [inline, virtual]

Creates a view of this byte buffer as a short buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new short Buffer, which the caller then owns.

Implements **decaf::nio::ByteBuffer** (p. 742).

6.115.3.10 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ByteArrayBuffer** (p. 694)

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 742).

6.115.3.11 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::duplicate () [virtual]

Creates a new byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new Byte Buffer which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 743).

6.115.3.12 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 743).

6.115.3.13 virtual unsigned char decaf::internal::nio::ByteArrayBuffer::get () const throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the byte at this buffer's current position, and then increments the position.

Returns:

The byte at the buffer's current position

Exceptions:

BufferUnderflowException - If the buffer's current position is not smaller than its limit

Implements **decaf::nio::ByteBuffer** (p. 744).

6.115.3.14 virtual char decaf::internal::nio::ByteArrayBuffer::getChar (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Reads one byte at the given index and returns it.

Parameters:

index - the index in the Buffer where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ByteBuffer** (p. 745).

6.115.3.15 virtual char decaf::internal::nio::ByteArrayBuffer::getChar () throw (decaf::nio::BufferUnderflowException) [inline, virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns:

the next char in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 745).

6.115.3.16 virtual double decaf::internal::nio::ByteArrayBuffer::getDouble (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 746).

6.115.3.17 virtual double decaf::internal::nio::ByteArrayBuffer::getDouble () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next double in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 746).

6.115.3.18 virtual float decaf::internal::nio::ByteArrayBuffer::getFloat (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 746).

6.115.3.19 virtual float decaf::internal::nio::ByteArrayBuffer::getFloat () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next float in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 747).

6.115.3.20 virtual int decaf::internal::nio::ByteArrayBuffer::getInt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 747).

6.115.3.21 virtual int decaf::internal::nio::ByteArrayBuffer::getInt () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next int in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 747).

6.115.3.22 virtual long long decaf::internal::nio::ByteArrayBuffer::getLong (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 748).

6.115.3.23 virtual long long decaf::internal::nio::ByteArrayBuffer::getLong () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next long long in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 748).

6.115.3.24 virtual short decaf::internal::nio::ByteArrayBuffer::getShort (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads two bytes at the given index and returns it.

Parameters:

index - the index in the Buffer where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implements **decaf::nio::ByteBuffer** (p. 748).

6.115.3.25 virtual short decaf::internal::nio::ByteArrayBuffer::getShort () throw (decaf::nio::BufferUnderflowException) [virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next short in the buffer..

Exceptions:

BufferUnderflowException - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implements **decaf::nio::ByteBuffer** (p. 749).

6.115.3.26 virtual bool decaf::internal::nio::ByteArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible byte array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::ByteBuffer** (p. 749).

6.115.3.27 virtual bool decaf::internal::nio::ByteArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 749).

6.115.3.28 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (std::size_t *index*, unsigned char *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given byte into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 750).

6.115.3.29 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::put (unsigned char *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters:

value - the byte value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 750).

6.115.3.30 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putChar (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 752).

6.115.3.31 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putChar (char *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 752).

6.115.3.32 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putDouble (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 753).

6.115.3.33 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putDouble (double *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 753).

6.115.3.34 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 754).

6.115.3.35 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putFloat (float *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 754).

6.115.3.36 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putInt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 754).

6.115.3.37 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putInt (int *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 755).

6.115.3.38 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putLong (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 755).

6.115.3.39 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putLong (long long *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 756).

6.115.3.40 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putShort (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 756).

6.115.3.41 virtual ByteArrayBuffer& decaf::internal::nio::ByteArrayBuffer::putShort (short *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ByteBuffer** (p. 756).

6.115.3.42 virtual void decaf::internal::nio::ByteArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.115.3.43 virtual ByteArrayBuffer* decaf::internal::nio::ByteArrayBuffer::slice () const [virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ByteArrayBuffer** (p. 694) which the caller owns.

Implements **decaf::nio::ByteBuffer** (p. 757).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ByteArrayBuffer.h**

6.116 decaf::io::ByteArrayInputStream Class Reference

Simple implementation of **InputStream** (p. 1406) that wraps around an STL Vector std::vector<unsigned char>.

`#include <src/main/decaf/io/ByteArrayInputStream.h>`Inheritance diagram for decaf::io::ByteArrayInputStream:

Public Member Functions

- **ByteArrayInputStream** ()

  Default Constructor.

- **ByteArrayInputStream** (const std::vector< unsigned char > &buffer)

  Creates the input stream and calls setBuffer with the specified buffer object.

- **ByteArrayInputStream** (const unsigned char *buffer, std::size_t bufferSize)

  Constructor.

- virtual ~**ByteArrayInputStream** ()
- virtual void **setBuffer** (const std::vector< unsigned char > &buffer)

  *Sets the **internal** (p. 95) buffer.*

- virtual void **setByteArray** (const unsigned char *buffer, std::size_t bufferSize)

  Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

- virtual std::size_t **available** () const throw (IOException)

  Indcates the number of bytes avaialable.

- virtual unsigned char **read** () throw (IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual void **close** () throw (lang::Exception)

  Closes the target input stream.

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit DECAF_UNUSED)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (IOException)

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

- virtual bool **markSupported** () const

  Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.116.1 Detailed Description

Simple implementation of **InputStream** (p. 1406) that wraps around an STL Vector std::vector<unsigned char>. Closing a **ByteArrayInputStream** (p. 716) has no effect. The methods in this class can be called after the stream has been closed without generating an **IOException** (p. 1477).

6.116.2 Constructor & Destructor Documentation

6.116.2.1 decaf::io::ByteArrayInputStream::ByteArrayInputStream ()

Default Constructor.

6.116.2.2 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const std::vector< unsigned char > & *buffer*)

Creates the input stream and calls setBuffer with the specified buffer object.

Parameters:

buffer The buffer to be used.

6.116.2.3 decaf::io::ByteArrayInputStream::ByteArrayInputStream (const unsigned char * *buffer*, std::size_t *bufferSize*)

Constructor.

Parameters:

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.116.2.4 virtual decaf::io::ByteArrayInputStream::~ByteArrayInputStream () [virtual]

6.116.3 Member Function Documentation

6.116.3.1 virtual std::size_t decaf::io::ByteArrayInputStream::available () const throw (IOException) [inline, virtual]

Indcates the number of bytes avaialable.

Returns:

The number of bytes until the end of the **internal** (p. 95) buffer.

Implements **decaf::io::InputStream** (p. 1407).

6.116.3.2 virtual void decaf::io::ByteArrayInputStream::close () throw (lang::Exception) [inline, virtual]

Closes the target input stream.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implements **decaf::io::Closeable** (p. 840).

6.116.3.3 virtual void decaf::io::ByteArrayInputStream::lock () throw (lang::Exception) [inline, protected, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.116.3.4 virtual void decaf::io::ByteArrayInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream

instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1407).

6.116.3.5 virtual bool decaf::io::ByteArrayInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1407).

6.116.3.6 virtual void decaf::io::ByteArrayInputStream::notify () throw (lang::Exception) [inline, protected, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.116.3.7 virtual void decaf::io::ByteArrayInputStream::notifyAll () throw (lang::Exception) [inline, protected, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.116.3.8 virtual int decaf::io::ByteArrayInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read.

Exceptions:

IOException *(*p. ***1477)*** thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.116.3.9 virtual unsigned char decaf::io::ByteArrayInputStream::read () throw (IOException) [virtual]

Reads a single byte from the buffer.

Returns:

The next byte.

Exceptions:

IOException *(*p. ***1477)*** thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.116.3.10 virtual void decaf::io::ByteArrayInputStream::reset () throw (IOException) [virtual]

Resets the read index to the beginning of the byte array, unless mark has been called and the markLimit has not been exceeded, in which case the stream is reset to the marked position.

Implements **decaf::io::InputStream** (p. 1408).

6.116.3.11 virtual void decaf::io::ByteArrayInputStream::setBuffer (const std::vector< unsigned char > & *buffer*) [virtual]

Sets the **internal** (p. 95) buffer. The input stream will wrap around this buffer and will perform all read operations on it. The position will be reinitialized to the beginning of the specified buffer. This class will not own the given buffer - it is the caller's responsibility to free the memory of the given buffer as appropriate.

Parameters:

buffer The buffer to be used.

6.116.3.12 virtual void decaf::io::ByteArrayInputStream::setByteArray (const unsigned char * *buffer*, std::size_t *bufferSize*) [virtual]

Sets the data that this reader uses, replaces any existing data and resets to beginning of the buffer.

Parameters:

buffer initial byte array to use to read from

bufferSize the size of the buffer

6.116.3.13 virtual std::size_t decaf::io::ByteArrayInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1406) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total butes skipped

Exceptions:

IOException *(* p. ***1477)*** if an error occurs

Implements **decaf::io::InputStream** (p. 1409).

6.116.3.14 virtual void decaf::io::ByteArrayInputStream::unlock () throw (lang::Exception) [inline, protected, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.116.3.15 virtual void decaf::io::ByteArrayInputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, protected, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.116.3.16 virtual void decaf::io::ByteArrayInputStream::wait () throw (lang::Exception) [inline, protected, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayInputStream.h**

6.117 decaf::io::ByteArrayOutputStream Class Reference

`#include <src/main/decaf/io/ByteArrayOutputStream.h>`Inheritance diagram for decaf::io::ByteArrayOutputStream:

Public Member Functions

- **ByteArrayOutputStream** ()

  *Default Constructor - uses a default **internal** (p. 95) buffer.*

- **ByteArrayOutputStream** (std::vector< unsigned char > &buffer)

  Uses the given buffer as the target.

- virtual ~**ByteArrayOutputStream** ()
- virtual void **setBuffer** (std::vector< unsigned char > &buffer)

  *Sets the **internal** (p. 95) buffer.*

- virtual const unsigned char * **toByteArray** () const

  Get a snapshot of the data.

- virtual std::size_t **size** () const

  Get the Size of the Internal Buffer.

- virtual void **write** (unsigned char c) throw (IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (IOException)

  Invokes flush on the target output stream, has no affect.

- virtual void **reset** () throw (IOException)

  Clear current Stream contents.

- void **close** () throw (lang::Exception)

  Invokes close on the target output stream.

- std::string **toString** () const

  Converts the bytes in the buffer into a standard C++ string.

- void **writeTo** (**OutputStream** *out) const throw (IOException, lang::exceptions::NullPointerException)

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).

- virtual void **lock** () throw (lang::Exception)
  Locks the object.
- virtual void **unlock** () throw (lang::Exception)
  Unlocks the object.
- virtual void **wait** () throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.117.1 Constructor & Destructor Documentation

6.117.1.1 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream ()

Default Constructor - uses a default **internal** (p. 95) buffer.

6.117.1.2 decaf::io::ByteArrayOutputStream::ByteArrayOutputStream (std::vector< unsigned char > & *buffer*)

Uses the given buffer as the target. Calls setBuffer.

Parameters:

buffer the target buffer.

6.117.1.3 virtual decaf::io::ByteArrayOutputStream::~ByteArrayOutputStream () [inline, virtual]

6.117.2 Member Function Documentation

6.117.2.1 void decaf::io::ByteArrayOutputStream::close () throw (lang::Exception) [inline, virtual]

Invokes close on the target output stream.

Exceptions:

CMSException

Implements **decaf::io::Closeable** (p. 840).

6.117.2.2 virtual void decaf::io::ByteArrayOutputStream::flush () throw (IOException) [inline, virtual]

Invokes flush on the target output stream, has no affect.

Exceptions:

IOException *(p. 1477)*

Implements **decaf::io::OutputStream** (p. 1992).

6.117.2.3 virtual void decaf::io::ByteArrayOutputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.117.2.4 virtual void decaf::io::ByteArrayOutputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.117.2.5 virtual void decaf::io::ByteArrayOutputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.117.2.6 virtual void decaf::io::ByteArrayOutputStream::reset () throw (IOException) [virtual]

Clear current Stream contents.

Exceptions:

IOException *(p. 1477)*

6.117.2.7 virtual void decaf::io::ByteArrayOutputStream::setBuffer (std::vector< unsigned char > & *buffer*) [virtual]

Sets the **internal** (p. 95) buffer. This input stream will wrap around the given buffer and all writes will be performed directly on the buffer. This object does not retain control of the buffer's lifetime however - this is the job of the caller.

Parameters:

buffer The target buffer.

6.117.2.8 virtual std::size_t decaf::io::ByteArrayOutputStream::size () const [inline, virtual]

Get the Size of the Internal Buffer.

Returns:

size of the **internal** (p. 95) buffer

6.117.2.9 virtual const unsigned char* decaf::io::ByteArrayOutputStream::toByteArray () const [inline, virtual]

Get a snapshot of the data.

Returns:

pointer to the data

6.117.2.10 std::string decaf::io::ByteArrayOutputStream::toString () const

Converts the bytes in the buffer into a standard C++ string.

Returns:

a string containing the bytes in the buffer

6.117.2.11 virtual void decaf::io::ByteArrayOutputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.117.2.12 virtual void decaf::io::ByteArrayOutputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.117.2.13 virtual void decaf::io::ByteArrayOutputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.117.2.14 virtual void decaf::io::ByteArrayOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 1993).

6.117.2.15 virtual void decaf::io::ByteArrayOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException *(* p. ***1477)*** thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

6.117.2.16 virtual void decaf::io::ByteArrayOutputStream::write (unsigned char *c*) throw (IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException *(* p. ***1477)*** thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

6.117.2.17 void decaf::io::ByteArrayOutputStream::writeTo (OutputStream * *out*) const throw (IOException, lang::exceptions::NullPointerException)

Writes the complete contents of this byte array output stream to the specified output stream argument, as if by calling the output stream's write method using out.write(buf, 0, count).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**ByteArrayOutputStream.h**

6.118 decaf::internal::nio::ByteArrayPerspective Class Reference

This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective.

`#include <src/main/decaf/internal/nio/ByteArrayPerspective.h>`Inheritance diagram for decaf::internal::nio::ByteArrayPerspective:

Public Member Functions

- **ByteArrayPerspective** (std::size_t capacity)

  Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted.

- **ByteArrayPerspective** (unsigned char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a byte array object that wraps the given array.

- **ByteArrayPerspective** (char *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- **ByteArrayPerspective** (double *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- **ByteArrayPerspective** (float *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- **ByteArrayPerspective** (long long *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- **ByteArrayPerspective** (int *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- **ByteArrayPerspective** (short *array, std::size_t capacity, bool own=false) throw (lang::exceptions::NullPointerException)

  Creates a array object that wraps the given array.

- virtual ~**ByteArrayPerspective** ()
- **ByteArrayPerspective** * **takeRef** ()

  Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.

- void **returnRef** ()

  Returns a reference to this Perspective, when the count is zero it should be deleted.

- int **getReferences** () const

6.118.1 Detailed Description

This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective. Creating an instance of this class implicitly takes a reference to it, so a creator must return its ref before the count will be zero.

6.118.2 Constructor & Destructor Documentation

6.118.2.1 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (std::size_t *capacity*)

Creates a byte array object that is allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.118.2.2 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (unsigned char * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a byte array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.3 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (char * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.4 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (double * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.5 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (float * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.6 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (long long * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.7 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (int * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.8 decaf::internal::nio::ByteArrayPerspective::ByteArrayPerspective (short * *array*, std::size_t *capacity*, bool *own* = `false`) throw (lang::exceptions::NullPointerException)

Creates a array object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

capacity - size of the array, this is the limit we read and write to.

own - is this class now the owner of the pointer.

Exceptions:

NullPointerException if buffer is NULL

6.118.2.9 virtual decaf::internal::nio::ByteArrayPerspective::~ByteArrayPerspective () `[inline, virtual]`

6.118.3 Member Function Documentation

6.118.3.1 int decaf::internal::nio::ByteArrayPerspective::getReferences () const `[inline]`

Returns:

the current number of reference on this perspective

6.118.3.2 void decaf::internal::nio::ByteArrayPerspective::returnRef () [inline]

Returns a reference to this Perspective, when the count is zero it should be deleted.

6.118.3.3 ByteArrayPerspective* decaf::internal::nio::ByteArrayPerspective::takeRef () [inline]

Takes a reference to this Perspective and returns a pointer to this so that a caller can take a ref and get a ref in one call.

Returns:

this

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ByteArrayPerspective.h**

6.119 decaf::nio::ByteBuffer Class Reference

This class defines six categories of operations upon byte buffers:.

`#include <src/main/decaf/nio/ByteBuffer.h>`Inheritance diagram for decaf::nio::ByteBuffer:

Public Member Functions

- virtual ~**ByteBuffer** ()
- virtual std::string **toString** () const
- **ByteBuffer** & **get** (std::vector< unsigned char > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **ByteBuffer** & **get** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- **ByteBuffer** & **put** (**ByteBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the bytes remaining in the given source buffer into this buffer.

- **ByteBuffer** & **put** (const unsigned char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers bytes into this buffer from the given source array.

- **ByteBuffer** & **put** (std::vector< unsigned char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source byte array into this buffer.

- virtual bool **isReadOnly** () const =0

  Tells whether or not this buffer is read-only.

- virtual unsigned char * **array** ()=0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)

  Returns the byte array that backs this buffer.

- virtual std::size_t **arrayOffset** () const =0 throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException)

  Returns the offset within this buffer's backing array of the first element of the buffer.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible byte array.

- virtual **CharBuffer** * **asCharBuffer** () const =0

  Creates a view of this byte buffer as a char buffer.

- virtual **DoubleBuffer** * **asDoubleBuffer** () const =0

  Creates a view of this byte buffer as a double buffer.

- virtual **FloatBuffer** * **asFloatBuffer** () const =0

  Creates a view of this byte buffer as a float buffer.

- virtual **IntBuffer** * **asIntBuffer** () const =0

  Creates a view of this byte buffer as a int buffer.

- virtual **LongBuffer** * **asLongBuffer** () const =0

  Creates a view of this byte buffer as a long buffer.

- virtual **ShortBuffer** * **asShortBuffer** () const =0

  Creates a view of this byte buffer as a short buffer.

- virtual **ByteBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only byte buffer that shares this buffer's content.

- virtual **ByteBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **ByteBuffer** * **duplicate** ()=0

  Creates a new byte buffer that shares this buffer's content.

- virtual unsigned char **get** () const =0 throw (BufferUnderflowException)

  Relative get method.

- virtual unsigned char **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- virtual char **getChar** ()=0 throw (BufferUnderflowException)

  Reads the next byte at this buffer's current position, and then increments the position by one.

- virtual char **getChar** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads one byte at the given index and returns it.

- virtual double **getDouble** ()=0 throw (BufferUnderflowException)

  Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual double **getDouble** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given index and returns it.

- virtual float **getFloat** ()=0 throw (BufferUnderflowException)

  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual float **getFloat** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given index and returns it.

- virtual long long **getLong** ()=0 throw (BufferUnderflowException)

  Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

- virtual long long **getLong** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads eight bytes at the given index and returns it.

- virtual int **getInt** ()=0 throw (BufferUnderflowException)

  Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

- virtual int **getInt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads four bytes at the given index and returns it.

- virtual short **getShort** ()=0 throw (BufferUnderflowException)

  Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

- virtual short **getShort** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Reads two bytes at the given index and returns it.

- virtual **ByteBuffer** & **put** (unsigned char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given byte into this buffer at the current position, and then increments the position.

- virtual **ByteBuffer** & **put** (std::size_t index, unsigned char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given byte into this buffer at the given index.

- virtual **ByteBuffer** & **putChar** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

- virtual **ByteBuffer** & **putChar** (std::size_t index, char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes one byte containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putDouble** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putDouble** (std::size_t index, double value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putFloat** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putFloat** (std::size_t index, float value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putLong** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putLong** (std::size_t index, long long value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes eight bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putInt** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putInt** (std::size_t index, int value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes four bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** & **putShort** (short value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

- virtual **ByteBuffer** & **putShort** (std::size_t index, short value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes two bytes containing the given value, into this buffer at the given index.

- virtual **ByteBuffer** * **slice** () const =0

  Creates a new byte buffer whose content is a shared subsequence of this buffer's content.

- virtual int **compareTo** (const **ByteBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **ByteBuffer** &value) const
- virtual bool **operator==** (const **ByteBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **ByteBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ByteBuffer** * **allocate** (std::size_t capacity)

  Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

- static **ByteBuffer** * **wrap** (unsigned char *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **ByteBuffer** (p. 734).*

- static **ByteBuffer** * **wrap** (std::vector< unsigned char > &buffer)

  *Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 734).*

Protected Member Functions

- **ByteBuffer** (std::size_t capacity)

  *Creates a **ByteBuffer** (p. 734) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.119.1 Detailed Description

This class defines six categories of operations upon byte buffers:. 1. Absolute and relative get and put methods that read and write single bytes; 2. Relative bulk get methods that transfer contiguous sequences of bytes from this buffer into an array; 3. Relative bulk put methods that transfer contiguous sequences of bytes from a byte array or some other byte buffer into this buffer; 4. Absolute and relative get and put methods that read and write values of other primitive types, translating them to and from sequences of bytes in a particular byte order; 5. Methods for creating view buffers, which allow a byte buffer to be viewed as a buffer containing values of some other primitive type; and 6. Methods for compacting, duplicating, and slicing a byte buffer.

Byte buffers can be created either by allocation, which allocates space for the buffer's content, or by wrapping an existing byte array into a buffer.

Access to binary data:

This class defines methods for reading and writing values of all other primitive types, except boolean. Primitive values are translated to (or from) sequences of bytes according to the buffer's current byte order.

For access to heterogeneous binary data, that is, sequences of values of different types, this class defines a family of absolute and relative get and put methods for each type. For 32-bit floating-point values, for example, this class defines:

float **getFloat()** (p. 747) float getFloat(int index) void **putFloat(float f)** (p. 754) void putFloat(int index, float f)

Corresponding methods are defined for the types char, short, int, long, and double. The index parameters of the absolute get and put methods are in terms of bytes rather than of the type being read or written.

For access to homogeneous binary data, that is, sequences of values of the same type, this class defines methods that can create views of a given byte buffer. A view buffer is simply another buffer whose content is backed by the byte buffer. Changes to the byte buffer's content will be visible in

the view buffer, and vice versa; the two buffers' position, limit, and mark values are independent. The asFloatBuffer method, for example, creates an instance of the **FloatBuffer** (p. 1346) class that is backed by the byte buffer upon which the method is invoked. Corresponding view-creation methods are defined for the types char, short, int, long, and double.

View buffers have two important advantages over the families of type-specific get and put methods described above:

A view buffer is indexed not in terms of bytes but rather in terms of the type-specific size of its values;

A view buffer provides relative bulk get and put methods that can transfer contiguous sequences of values between a buffer and an array or some other buffer of the same type; and

6.119.2 Constructor & Destructor Documentation

6.119.2.1 decaf::nio::ByteBuffer::ByteBuffer (std::size_t *capacity*) [protected]

Creates a **ByteBuffer** (p. 734) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.119.2.2 virtual decaf::nio::ByteBuffer::~ByteBuffer () [inline, virtual]

6.119.3 Member Function Documentation

6.119.3.1 static ByteBuffer* decaf::nio::ByteBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new byte buffer whose position will be zero its limit will be its capacity and its mark is not set.

Parameters:

capacity - the **internal** (p. 95) buffer's capacity.

Returns:

a newly allocated **ByteBuffer** (p. 734) which the caller owns.

6.119.3.2 virtual unsigned char* decaf::nio::ByteBuffer::array () throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns the byte array that backs this buffer. Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The array that backs this buffer

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 700).

6.119.3.3 virtual std::size_t decaf::nio::ByteBuffer::arrayOffset () const throw (ReadOnlyBufferException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer. If this buffer is backed by an array then buffer position p corresponds to array index p + **arrayOffset()** (p. 740).

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset within this buffer's array of the first element of the buffer

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is backed by an array but is read-only

UnsupportedOperationException - If this buffer is not backed by an accessible array

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 700).

6.119.3.4 virtual CharBuffer* decaf::nio::ByteBuffer::asCharBuffer () const [pure virtual]

Creates a view of this byte buffer as a char buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new Char **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 700).

6.119.3.5 virtual DoubleBuffer* decaf::nio::ByteBuffer::asDoubleBuffer () const [pure virtual]

Creates a view of this byte buffer as a double buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new double **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 701).

6.119.3.6 virtual FloatBuffer* decaf::nio::ByteBuffer::asFloatBuffer () const [pure virtual]

Creates a view of this byte buffer as a float buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new float **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 701).

6.119.3.7 virtual IntBuffer* decaf::nio::ByteBuffer::asIntBuffer () const [pure virtual]

Creates a view of this byte buffer as a int buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by four, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new int **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 701).

6.119.3.8 virtual LongBuffer* decaf::nio::ByteBuffer::asLongBuffer () const [pure virtual]

Creates a view of this byte buffer as a long buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by eight, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new long **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 702).

6.119.3.9 virtual ByteBuffer* decaf::nio::ByteBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only byte buffer which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 702).

6.119.3.10 virtual ShortBuffer* decaf::nio::ByteBuffer::asShortBuffer () const [pure virtual]

Creates a view of this byte buffer as a short buffer. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer divided by two, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the new short **Buffer** (p. 627), which the caller then owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 702).

6.119.3.11 virtual ByteBuffer& decaf::nio::ByteBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ByteBuffer** (p. 734)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 703).

6.119.3.12 virtual int decaf::nio::ByteBuffer::compareTo (const ByteBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.119.3.13 virtual ByteBuffer* decaf::nio::ByteBuffer::duplicate () [pure virtual]

Creates a new byte buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new Byte **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 703).

6.119.3.14 virtual bool decaf::nio::ByteBuffer::equals (const ByteBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.119.3.15 virtual unsigned char decaf::nio::ByteBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the byte at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the byte is to be read

Returns:

the byte that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 703).

6.119.3.16 virtual unsigned char decaf::nio::ByteBuffer::get () const throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the byte at this buffer's current position, and then increments the position.

Returns:

The byte at the buffer's current position

Exceptions:

BufferUnderflowException (p. ***661***) - If the buffer's current position is not smaller than its limit

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 704).

6.119.3.17 ByteBuffer& decaf::nio::ByteBuffer::get (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers bytes from this buffer into the given destination array. If there are fewer bytes remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length bytes from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException (p. ***661***) - If there are fewer than length bytes remaining in this buffer

NullPointerException if the passed buffer is null.

6.119.3.18 ByteBuffer& decaf::nio::ByteBuffer::get (std::vector< unsigned char > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers bytes from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this Byte **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length bytes remaining in this buffer

6.119.3.19 virtual char decaf::nio::ByteBuffer::getChar (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads one byte at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the byte is to be read

Returns:

the char at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 704).

6.119.3.20 virtual char decaf::nio::ByteBuffer::getChar () throw (BufferUnderflowException) [pure virtual]

Reads the next byte at this buffer's current position, and then increments the position by one.

Returns:

the next char in the buffer..

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 704).

6.119.3.21 virtual double decaf::nio::ByteBuffer::getDouble (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the bytes are to be read

Returns:

the double at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 705).

6.119.3.22 virtual double decaf::nio::ByteBuffer::getDouble () throw (BufferUnderflowException) [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next double in the buffer..

Exceptions:

BufferUnderflowException *(p. 661)* - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 705).

6.119.3.23 virtual float decaf::nio::ByteBuffer::getFloat (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the bytes are to be read

Returns:

the float at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 705).

6.119.3.24 virtual float decaf::nio::ByteBuffer::getFloat () throw (BufferUnderflowException) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next float in the buffer..

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 706).

6.119.3.25 virtual int decaf::nio::ByteBuffer::getInt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads four bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the bytes are to be read

Returns:

the int at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 706).

6.119.3.26 virtual int decaf::nio::ByteBuffer::getInt () throw (BufferUnderflowException) [pure virtual]

Reads the next four bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next int in the buffer..

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 707).

6.119.3.27 virtual long long decaf::nio::ByteBuffer::getLong (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads eight bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the bytes are to be read

Returns:

the long long at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 707).

6.119.3.28 virtual long long decaf::nio::ByteBuffer::getLong () throw (BufferUnderflowException) [pure virtual]

Reads the next eight bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next long long in the buffer..

Exceptions:

BufferUnderflowException *(p. 661)* - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 707).

6.119.3.29 virtual short decaf::nio::ByteBuffer::getShort (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Reads two bytes at the given index and returns it.

Parameters:

index - the index in the **Buffer** (p. 627) where the bytes are to be read

Returns:

the short at the given index in the buffer

Exceptions:

IndexOutOfBoundsException - If there are not enough bytes remaining to fill the requested Data Type

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 708).

6.119.3.30 virtual short decaf::nio::ByteBuffer::getShort () throw (BufferUnderflowException) [pure virtual]

Reads the next two bytes at this buffer's current position, and then increments the position by that amount.

Returns:

the next short in the buffer..

Exceptions:

BufferUnderflowException *(*p. *661)* - If there are no more bytes remaining in this buffer, meaning we have reached the set limit.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 708).

6.119.3.31 virtual bool decaf::nio::ByteBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible byte array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 708).

6.119.3.32 virtual bool decaf::nio::ByteBuffer::isReadOnly () const [pure virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 708).

6.119.3.33 virtual bool decaf::nio::ByteBuffer::operator< (const ByteBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.119.3.34 virtual bool decaf::nio::ByteBuffer::operator== (const ByteBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.119.3.35 virtual ByteBuffer& decaf::nio::ByteBuffer::put (std::size_t *index*, unsigned char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given byte into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the byte to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 709).

6.119.3.36 virtual ByteBuffer& decaf::nio::ByteBuffer::put (unsigned char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given byte into this buffer at the current position, and then increments the position.

Parameters:

value - the byte value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 709).

6.119.3.37 ByteBuffer& decaf::nio::ByteBuffer::put (std::vector< unsigned char > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source byte array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **ByteBuffer** (p. 734)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

6.119.3.38 ByteBuffer& decaf::nio::ByteBuffer::put (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers bytes into this buffer from the given source array. If there are more bytes to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer - The array from which bytes are to be read

offset - The offset within the array of the first byte to be read;

length - The number of bytes to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.119.3.39 ByteBuffer& decaf::nio::ByteBuffer::put (ByteBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the bytes remaining in the given source buffer into this buffer. If there are more bytes remaining in the source buffer than in this buffer, that is, if src.remaining() >

remaining() (p. 632), then no bytes are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() bytes from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take bytes from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer for the remaining bytes in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.119.3.40 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes one byte containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 710).

6.119.3.41 virtual ByteBuffer& decaf::nio::ByteBuffer::putChar (char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes one byte containing the given value, into this buffer at the current position, and then increments the position by one.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 710).

6.119.3.42 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 710).

6.119.3.43 virtual ByteBuffer& decaf::nio::ByteBuffer::putDouble (double *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 711).

6.119.3.44 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 711).

6.119.3.45 virtual ByteBuffer& decaf::nio::ByteBuffer::putFloat (float *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 712).

6.119.3.46 virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes four bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 712).

6.119.3.47 virtual ByteBuffer& decaf::nio::ByteBuffer::putInt (int *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes four bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 713).

6.119.3.48 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 713).

6.119.3.49 virtual ByteBuffer& decaf::nio::ByteBuffer::putLong (long long *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes eight bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 713).

6.119.3.50 virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes two bytes containing the given value, into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the value to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 714).

6.119.3.51 virtual ByteBuffer& decaf::nio::ByteBuffer::putShort (short *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes two bytes containing the given value, into this buffer at the current position, and then increments the position by eight.

Parameters:

value - The value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. *658)* - If there are fewer than bytes remaining in this buffer than the size of the data to be written

ReadOnlyBufferException *(*p. *2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 714).

6.119.3.52 virtual ByteBuffer* decaf::nio::ByteBuffer::slice () const [pure virtual]

Creates a new byte buffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ByteBuffer** (p. 734) which the caller owns.

Implemented in **decaf::internal::nio::ByteArrayBuffer** (p. 715).

6.119.3.53 virtual std::string decaf::nio::ByteBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.119.3.54 static ByteBuffer* decaf::nio::ByteBuffer::wrap (std::vector< unsigned char > & *buffer*) [static]

Wraps the passed STL Byte Vector in a **ByteBuffer** (p. 734). The new buffer will be backed by the given byte array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **ByteBuffer** (p. 734) that is backed by buffer, caller owns.

6.119.3.55 static ByteBuffer* decaf::nio::ByteBuffer::wrap (unsigned char * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **ByteBuffer** (p. 734). The new buffer will be backed by the given byte array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **ByteBuffer** (p. 734) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ByteBuffer.h**

6.120 cms::BytesMessage Class Reference

A **BytesMessage** (p. 759) object is used to send a message containing a stream of unsigned bytes.

`#include <src/main/cms/BytesMessage.h>`Inheritance diagram for cms::BytesMessage:

Public Member Functions

- virtual ~**BytesMessage** ()
- virtual void **setBodyBytes** (const unsigned char *buffer, std::size_t numBytes)=0 throw (CMSException)

  sets the bytes given to the message body.

- virtual const unsigned char * **getBodyBytes** () const =0 throw (CMSException)

  Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer.

- virtual std::size_t **getBodyLength** () const =0 throw (CMSException)

  Returns the number of bytes contained in the body of this message.

- virtual void **reset** ()=0 throw (cms::CMSException)

  Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

- virtual bool **readBoolean** () const =0 throw (cms::CMSException)

  Reads a Boolean from the Bytes message stream.

- virtual void **writeBoolean** (bool value)=0 throw (cms::CMSException)

  Writes a boolean to the bytes message stream as a 1-byte value.

- virtual unsigned char **readByte** () const =0 throw (cms::CMSException)

  Reads a Byte from the Bytes message stream.

- virtual void **writeByte** (unsigned char value)=0 throw (cms::CMSException)

  Writes a byte to the bytes message stream as a 1-byte value.

- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::CMSException)

  Reads a byte array from the bytes message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::CMSException)

  Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *&buffer, std::size_t length) const =0 throw (cms::CMSException)

  Reads a portion of the bytes message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length)=0 throw (cms::CMSException)

  Writes a portion of a byte array to the bytes message stream.

- virtual char **readChar** () const =0 throw (cms::CMSException)

  Reads a Char from the Bytes message stream.

- virtual void **writeChar** (char value)=0 throw (cms::CMSException)

  Writes a char to the bytes message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::CMSException)

  Reads a 32 bit float from the Bytes message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::CMSException)

  Writes a float to the bytes message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::CMSException)

  Reads a 64 bit double from the Bytes message stream.

- virtual void **writeDouble** (double value)=0 throw (cms::CMSException)

  Writes a double to the bytes message stream as a 8 byte value.

- virtual short **readShort** () const =0 throw (cms::CMSException)

  Reads a 16 bit signed short from the Bytes message stream.

- virtual void **writeShort** (short value)=0 throw (cms::CMSException)

  Writes a signed short to the bytes message stream as a 2 byte value.

- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::CMSException)

  Reads a 16 bit unsigned short from the Bytes message stream.

- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::CMSException)

  Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual int **readInt** () const =0 throw (cms::CMSException)

  Reads a 32 bit signed integer from the Bytes message stream.

- virtual void **writeInt** (int value)=0 throw (cms::CMSException)

  Writes a signed int to the bytes message stream as a 4 byte value.

- virtual long long **readLong** () const =0 throw (cms::CMSException)

  Reads a 64 bit long from the Bytes message stream.

- virtual void **writeLong** (long long value)=0 throw (cms::CMSException)

  Writes a long long to the bytes message stream as a 8 byte value.

- virtual std::string **readString** () const =0 throw (cms::CMSException)

  Reads an ASCII String from the Bytes message stream.

- virtual void **writeString** (const std::string &value)=0 throw (cms::CMSException)

  Writes an ASCII String to the Bytes message stream.

- virtual std::string **readUTF** () const =0 throw (cms::CMSException)

  *Reads an UTF String from the **BytesMessage** (p. 759) stream.*

- virtual void **writeUTF** (const std::string &value)=0 throw (cms::CMSException)

  *Writes an UTF String to the **BytesMessage** (p. 759) stream.*

- virtual **BytesMessage** * **clone** () const =0

  Clones this message.

6.120.1 Detailed Description

A **BytesMessage** (p. 759) object is used to send a message containing a stream of unsigned bytes. It inherits from the **Message** (p. 1753) interface and adds a bytes message body. The receiver of the message supplies the interpretation of the bytes using the methods added by the **BytesMessage** (p. 759) interface.

The **BytesMessage** (p. 759) methods are based largely on those found in **decaf.io.DataInputStream** (p. 1116) and **decaf.io.DataOutputStream** (p. 1125).

Although the CMS API allows the use of message properties with byte messages, they are typically not used, since the inclusion of properties may affect the format.

The primitive types can be written explicitly using methods for each type. Because the C++ language is more limited when dealing with primitive types the JMS equivalent generic read and write methods that take Java objects cannot be provided in the CMS API.

When the message is first created, and when clearBody is called, the body of the message is in write-only mode. After the first call to reset has been made, the message body is in read-only mode. After a message has been sent, the client that sent it can retain and modify it without affecting the message that has been sent. The same message object can be sent multiple times. When a message has been received, the provider has called reset so that the message body is in read-only mode for the client.

If clearBody is called on a message in read-only mode, the message body is cleared and the message is in write-only mode.

If a client attempts to read a message in write-only mode, a **MessageNotReadableException** (p. 1876) is thrown.

If a client attempts to write a message in read-only mode, a **MessageNotWriteableException** (p. 1877) is thrown.

Since:

1.0

6.120.2 Constructor & Destructor Documentation

6.120.2.1 virtual cms::BytesMessage::~BytesMessage () [inline, virtual]

6.120.3 Member Function Documentation

6.120.3.1 virtual BytesMessage* cms::BytesMessage::clone () const [pure virtual]

Clones this message.

Returns:

a deep copy of this message.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs while cloning the **Message** (p. 1753).

Implements **cms::Message** (p. 1758).

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 166).

6.120.3.2 virtual const unsigned char* cms::BytesMessage::getBodyBytes () const throw (CMSException) [pure virtual]

Gets the bytes that are contained in this message, user should copy this data into a user allocated buffer. Call `getBodyLength` to determine the number of bytes to expect.

Returns:

const pointer to a byte buffer

Exceptions:

CMSException *(*p. ***850****)* - If an internal error occurs.

MessageNotReadableException *(*p. ***1876****)* - If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 167).

6.120.3.3 virtual std::size_t cms::BytesMessage::getBodyLength () const throw (CMSException) [pure virtual]

Returns the number of bytes contained in the body of this message.

Returns:

number of bytes.

Exceptions:

CMSException *(*p. ***850****)* - If an internal error occurs.

MessageNotReadableException *(*p. ***1876****)* - If the message is in Write Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 167).

6.120.3.4 virtual bool cms::BytesMessage::readBoolean () const throw (cms::CMSException) [pure virtual]

Reads a Boolean from the Bytes message stream.

Returns:

boolean value from stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 168).

6.120.3.5 virtual unsigned char cms::BytesMessage::readByte () const throw (cms::CMSException) [pure virtual]

Reads a Byte from the Bytes message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 168).

6.120.3.6 virtual std::size_t cms::BytesMessage::readBytes (unsigned char *& *buffer*, std::size_t *length*) const throw (cms::CMSException) [pure virtual]

Reads a portion of the bytes message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an IndexOutOfBoundsException is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 168).

6.120.3.7 virtual std::size_t cms::BytesMessage::readBytes (std::vector< unsigned char > & *value*) const throw (cms::CMSException) [pure virtual]

Reads a byte array from the bytes message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 169).

6.120.3.8 virtual char cms::BytesMessage::readChar () const throw (cms::CMSException) [pure virtual]

Reads a Char from the Bytes message stream.

Returns:

char value from stream

Exceptions:

CMSException *(*p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 169).

6.120.3.9 virtual double cms::BytesMessage::readDouble () const throw (cms::CMSException) [pure virtual]

Reads a 64 bit double from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException *(*p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 170).

6.120.3.10 virtual float cms::BytesMessage::readFloat () const throw (cms::CMSException) [pure virtual]

Reads a 32 bit float from the Bytes message stream.

Returns:

double value from stream

Exceptions:

CMSException *(*p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 170).

6.120.3.11 virtual int cms::BytesMessage::readInt () const throw (cms::CMSException) [pure virtual]

Reads a 32 bit signed integer from the Bytes message stream.

Returns:

int value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 170).

6.120.3.12 virtual long long cms::BytesMessage::readLong () const throw (cms::CMSException) [pure virtual]

Reads a 64 bit long from the Bytes message stream.

Returns:

long long value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 171).

6.120.3.13 virtual short cms::BytesMessage::readShort () const throw (cms::CMSException) [pure virtual]

Reads a 16 bit signed short from the Bytes message stream.

Returns:

short value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 171).

6.120.3.14 virtual std::string cms::BytesMessage::readString () const throw (cms::CMSException) [pure virtual]

Reads an ASCII String from the Bytes message stream.

Returns:

String from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 171).

6.120.3.15 virtual unsigned short cms::BytesMessage::readUnsignedShort () const throw (cms::CMSException) [pure virtual]

Reads a 16 bit unsigned short from the Bytes message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 171).

6.120.3.16 virtual std::string cms::BytesMessage::readUTF () const throw (cms::CMSException) [pure virtual]

Reads an UTF String from the **BytesMessage** (p. 759) stream.

Returns:

String from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of bytes stream has been reached.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 172).

6.120.3.17 virtual void cms::BytesMessage::reset () throw (cms::CMSException) [pure virtual]

Puts the message body in read-only mode and repositions the stream of bytes to the beginning.

Exceptions:

CMSException *(*p. ***850****)* - If the provider fails to perform the reset operation.

MessageFormatException *(p. 1842)* - If the **Message** (p. 1753) has an invalid format.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 172).

6.120.3.18 virtual void cms::BytesMessage::setBodyBytes (const unsigned char * *buffer*, std::size_t *numBytes*) throw (CMSException) [pure virtual]

sets the bytes given to the message body.

Parameters:

buffer Byte Buffer to copy

numBytes Number of bytes in Buffer to copy

Exceptions:

CMSException *(p. 850)* - If an internal error occurs.

MessageNotWriteableException *(p. 1877)* - if in Read Only Mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 172).

6.120.3.19 virtual void cms::BytesMessage::writeBoolean (bool *value*) throw (cms::CMSException) [pure virtual]

Writes a boolean to the bytes message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSException *(p. 850)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(p. 1877)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 173).

6.120.3.20 virtual void cms::BytesMessage::writeByte (unsigned char *value*) throw (cms::CMSException) [pure virtual]

Writes a byte to the bytes message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException *(p. 850)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(p. 1877)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 173).

6.120.3.21 virtual void cms::BytesMessage::writeBytes (const unsigned char * *value*, std::size_t *offset*, std::size_t *length*) throw (cms::CMSException) [pure virtual]

Writes a portion of a byte array to the bytes message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 173).

6.120.3.22 virtual void cms::BytesMessage::writeBytes (const std::vector< unsigned char > & *value*) throw (cms::CMSException) [pure virtual]

Writes a byte array to the bytes message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 174).

6.120.3.23 virtual void cms::BytesMessage::writeChar (char *value*) throw (cms::CMSException) [pure virtual]

Writes a char to the bytes message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 174).

6.120.3.24 virtual void cms::BytesMessage::writeDouble (double *value*) throw (cms::CMSException) [pure virtual]

Writes a double to the bytes message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 174).

6.120.3.25 virtual void cms::BytesMessage::writeFloat (float *value*) throw (cms::CMSException) [pure virtual]

Writes a float to the bytes message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.120.3.26 virtual void cms::BytesMessage::writeInt (int *value*) throw (cms::CMSException) [pure virtual]

Writes a signed int to the bytes message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.120.3.27 virtual void cms::BytesMessage::writeLong (long long *value*) throw (cms::CMSException) [pure virtual]

Writes a long long to the bytes message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.120.3.28 virtual void cms::BytesMessage::writeShort (short *value*) throw (cms::CMSException) [pure virtual]

Writes a signed short to the bytes message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 175).

6.120.3.29 virtual void cms::BytesMessage::writeString (const std::string & *value*) throw (cms::CMSException) [pure virtual]

Writes an ASCII String to the Bytes message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

6.120.3.30 virtual void cms::BytesMessage::writeUnsignedShort (unsigned short *value*) throw (cms::CMSException) [pure virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

6.120.3.31 virtual void cms::BytesMessage::writeUTF (const std::string & *value*) throw (cms::CMSException) [pure virtual]

Writes an UTF String to the **BytesMessage** (p. 759) stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of bytes stream has been reached.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 176).

The documentation for this class was generated from the following file:

- src/main/cms/**BytesMessage.h**

6.121 activemq::cmsutil::CachedConsumer Class Reference

A cached message consumer contained within a pooled session.

`#include <src/main/activemq/cmsutil/CachedConsumer.h>`Inheritance diagram for activemq::cmsutil::CachedConsumer:

Public Member Functions

- **CachedConsumer** (**cms::MessageConsumer** *consumer)
- virtual ~**CachedConsumer** ()
- virtual void **close** () throw (cms::CMSException)

  Does nothing - the real producer resource will be closed by the lifecycle manager.

- virtual **cms::Message** * **receive** () throw (cms::CMSException)

  Synchronously Receive a Message.

- virtual **cms::Message** * **receive** (int millisecs) throw (cms::CMSException)

  Synchronously Receive a Message, time out after defined interval.

- virtual **cms::Message** * **receiveNoWait** () throw (cms::CMSException)

  Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

- virtual void **setMessageListener** (**cms::MessageListener** *listener) throw (cms::CMSException)

  Sets the MessageListener that this class will send notifs on.

- virtual **cms::MessageListener** * **getMessageListener** () const throw (cms::CMSException)

  Gets the MessageListener that this class will send mew Message notification events to.

- virtual std::string **getMessageSelector** () const throw (cms::CMSException)

  Gets this message consumer's message selector expression.

6.121.1 Detailed Description

A cached message consumer contained within a pooled session.

6.121.2 Constructor & Destructor Documentation

6.121.2.1 activemq::cmsutil::CachedConsumer::CachedConsumer (cms::MessageConsumer * *consumer*) [inline]

6.121.2.2 virtual activemq::cmsutil::CachedConsumer::~CachedConsumer () [inline, virtual]

6.121.3 Member Function Documentation

6.121.3.1 virtual void activemq::cmsutil::CachedConsumer::close () throw (cms::CMSException) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 838).

6.121.3.2 virtual cms::MessageListener* activemq::cmsutil::CachedConsumer::getMessageListener () const throw (cms::CMSException) [inline, virtual]

Gets the MessageListener that this class will send mew Message notification events to.

Returns:

The listener of messages received by this consumer

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1796).

6.121.3.3 virtual std::string activemq::cmsutil::CachedConsumer::getMessageSelector () const throw (cms::CMSException) [inline, virtual]

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1796).

6.121.3.4 virtual cms::Message* activemq::cmsutil::CachedConsumer::receive (int *millisecs*) throw (cms::CMSException) [inline, virtual]

Synchronously Receive a Message, time out after defined interval. Returns null if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1796).

6.121.3.5 virtual cms::Message* activemq::cmsutil::CachedConsumer::receive () throw (cms::CMSException) [inline, virtual]

Synchronously Receive a Message.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1797).

6.121.3.6 virtual cms::Message* activemq::cmsutil::CachedConsumer::receiveNoWait () throw (cms::CMSException) [inline, virtual]

Receive a Message, does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1797).

6.121.3.7 virtual void activemq::cmsutil::CachedConsumer::setMessageListener (cms::MessageListener * *listener*) throw (cms::CMSException) [inline, virtual]

Sets the MessageListener that this class will send notifs on.

Parameters:

listener The listener of messages received by this consumer.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::MessageConsumer** (p. 1797).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedConsumer.h**

6.122 activemq::cmsutil::CachedProducer Class Reference

A cached message producer contained within a pooled session.

`#include <src/main/activemq/cmsutil/CachedProducer.h>`Inheritance diagram for activemq::cmsutil::CachedProducer:

Public Member Functions

- **CachedProducer** (**cms::MessageProducer** *producer)
- virtual ~**CachedProducer** ()
- virtual void **close** () throw (cms::CMSException)

  Does nothing - the real producer resource will be closed by the lifecycle manager.

- virtual void **send** (**cms::Message** *message) throw (cms::CMSException)

  Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (**cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException)

  Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message) throw (cms::CMSException)

  Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **cms::Destination** *destination, **cms::Message** *message, int deliveryMode, int priority, long long timeToLive) throw (cms::CMSException)

  Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode) throw (cms::CMSException)

  Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const throw (cms::CMSException)

  Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value) throw (cms::CMSException)

  Sets if Message Ids are disabled for this Producer.

- virtual bool **getDisableMessageID** () const throw (cms::CMSException)

  Gets if Message Ids are disabled for this Producer.

- virtual void **setDisableMessageTimeStamp** (bool value) throw (cms::CMSException)

  Sets if Message Time Stamps are disabled for this Producer.

- virtual bool **getDisableMessageTimeStamp** () const throw (cms::CMSException)

Gets if Message Time Stamps are disabled for this Producer.

- virtual void **setPriority** (int priority) throw (cms::CMSException)

  Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const throw (cms::CMSException)

  Gets the Priority level that this producer sends messages at.

- virtual void **setTimeToLive** (long long time) throw (cms::CMSException)

  Sets the Time to Live that this Producers sends messages with.

- virtual long long **getTimeToLive** () const throw (cms::CMSException)

  Gets the Time to Live that this producer sends messages with.

6.122.1 Detailed Description

A cached message producer contained within a pooled session.

6.122.2 Constructor & Destructor Documentation

6.122.2.1 activemq::cmsutil::CachedProducer::CachedProducer (cms::MessageProducer * *producer*) [inline]

6.122.2.2 virtual activemq::cmsutil::CachedProducer::~CachedProducer () [inline, virtual]

6.122.3 Member Function Documentation

6.122.3.1 virtual void activemq::cmsutil::CachedProducer::close () throw (cms::CMSException) [inline, virtual]

Does nothing - the real producer resource will be closed by the lifecycle manager.

Implements **cms::Closeable** (p. 838).

6.122.3.2 virtual int activemq::cmsutil::CachedProducer::getDeliveryMode () const throw (cms::CMSException) [inline, virtual]

Gets the delivery mode for this Producer.

Returns:

The DeliveryMode

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1879).

6.122.3.3 virtual bool activemq::cmsutil::CachedProducer::getDisableMessageID () const throw (cms::CMSException) [inline, virtual]

Gets if Message Ids are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1880).

6.122.3.4 virtual bool activemq::cmsutil::CachedProducer::getDisableMessageTimeStamp () const throw (cms::CMSException) [inline, virtual]

Gets if Message Time Stamps are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1880).

6.122.3.5 virtual int activemq::cmsutil::CachedProducer::getPriority () const throw (cms::CMSException) [inline, virtual]

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1880).

6.122.3.6 virtual long long activemq::cmsutil::CachedProducer::getTimeToLive () const throw (cms::CMSException) [inline, virtual]

Gets the Time to Live that this producer sends messages with.

Returns:

Time to live value in milliseconds

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1880).

6.122.3.7 virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * *destination*, cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSException) [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1881).

6.122.3.8 virtual void activemq::cmsutil::CachedProducer::send (const cms::Destination * *destination*, cms::Message * *message*) throw (cms::CMSException) [inline, virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

destination The destination on which to send the message

message the message to be sent.

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1881).

6.122.3.9 virtual void activemq::cmsutil::CachedProducer::send (cms::Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (cms::CMSException) [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1882).

6.122.3.10 virtual void activemq::cmsutil::CachedProducer::send (cms::Message * *message*) throw (cms::CMSException) [inline, virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

message The message to be sent.

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1882).

6.122.3.11 virtual void activemq::cmsutil::CachedProducer::setDeliveryMode (int *mode*) throw (cms::CMSException) [inline, virtual]

Sets the delivery mode for this Producer.

Parameters:

mode The DeliveryMode

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1882).

6.122.3.12 virtual void activemq::cmsutil::CachedProducer::setDisableMessageID (bool *value*) throw (cms::CMSException) [inline, virtual]

Sets if Message Ids are disabled for this Producer.

Parameters:

value boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1883).

6.122.3.13 virtual void activemq::cmsutil::CachedProducer::setDisableMessageTimeStamp (bool *value*) throw (cms::CMSException) [inline, virtual]

Sets if Message Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1883).

6.122.3.14 virtual void activemq::cmsutil::CachedProducer::setPriority (int *priority*) throw (cms::CMSException) [inline, virtual]

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1883).

6.122.3.15 virtual void activemq::cmsutil::CachedProducer::setTimeToLive (long long *time*) throw (cms::CMSException) [inline, virtual]

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

Parameters:

time default time to live value in milliseconds

Exceptions:

CMSException - if an internal error occurs.

Implements **cms::MessageProducer** (p. 1883).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CachedProducer.h**

6.123 decaf::util::concurrent::Callable< V > Class Template Reference

`#include <src/main/decaf/util/concurrent/Callable.h>`

Public Member Functions

- virtual ~**Callable** ()
- virtual V **call** ()=0 throw (decaf::lang::Exception)

  Computes a result, or throws an exception if unable to do so.

template<typename V> class decaf::util::concurrent::Callable< V >

6.123.1 Constructor & Destructor Documentation

6.123.1.1 template<typename V > virtual decaf::util::concurrent::Callable< V >::~Callable () [inline, virtual]

6.123.2 Member Function Documentation

6.123.2.1 template<typename V > virtual V decaf::util::concurrent::Callable< V >::call () throw (decaf::lang::Exception) [pure virtual]

Computes a result, or throws an exception if unable to do so.

Returns:

Computed Result.

Exceptions:

Exception If unable to compute a result.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Callable.h**

6.124 decaf::util::concurrent::CancellationException Class Reference

`#include <src/main/decaf/util/concurrent/CancellationException.h>`Inheritance diagram for decaf::util::concurrent::CancellationException:

Public Member Functions

- **CancellationException** () throw ()
  Default Constructor.
- **CancellationException** (const **decaf::lang::Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **CancellationException** (const **CancellationException** &ex) throw ()
  Copy Constructor.
- **CancellationException** (const std::exception ∗**cause**) throw ()
  Constructor.
- **CancellationException** (const char ∗file, const int lineNumber, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **CancellationException** (const char ∗file, const int lineNumber, const std::exception ∗**cause**, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **CancellationException** ∗ **clone** () const
  Clones this exception.
- virtual ∼**CancellationException** () throw ()

6.124.1 Constructor & Destructor Documentation

6.124.1.1 decaf::util::concurrent::CancellationException::CancellationException () throw () [inline]

Default Constructor.

6.124.1.2 decaf::util::concurrent::CancellationException::CancellationException (const decaf::lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.124.1.3 decaf::util::concurrent::CancellationException::CancellationException (const CancellationException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.124.1.4 decaf::util::concurrent::CancellationException::CancellationException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.124.1.5 decaf::util::concurrent::CancellationException::CancellationException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.124.1.6 decaf::util::concurrent::CancellationException::CancellationException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.124.1.7 virtual decaf::util::concurrent::CancellationException::~CancellationException () throw () [inline, virtual]

6.124.2 Member Function Documentation

6.124.2.1 virtual CancellationException* decaf::util::concurrent::CancellationException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CancellationException.h**

6.125 decaf::security::cert::Certificate Class Reference

Base interface for all identity certificates.

`#include <src/main/decaf/security/cert/Certificate.h>`Inheritance diagram for decaf::security::cert::Certificate:

Public Member Functions

- virtual ~**Certificate** ()
- virtual bool **equals** (const **Certificate** &cert) const =0

  Compares the encoded form of the two certificates.

- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)

  Provides the encoded form of this certificate.

- virtual std::string **getType** () const =0

  Returns the type of this certificate.

- virtual **PublicKey** * **getPublicKey** ()=0

  Gets the public key of this certificate.

- virtual const **PublicKey** * **getPublicKey** () const =0

  Gets the public key of this certificate.

- virtual void **verify** (const **PublicKey** &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)

  Verifies that this certificate was signed with the private key that corresponds to the specified public key.

- virtual void **verify** (const **PublicKey** &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)

  Verifies that this certificate was signed with the private key that corresponds to the specified public key.

- virtual std::string **toString** () const =0

  Returns a string representation of this certificate.

6.125.1 Detailed Description

Base interface for all identity certificates.

6.125.2 Constructor & Destructor Documentation

6.125.2.1 virtual decaf::security::cert::Certificate::~Certificate () [inline, virtual]

6.125.3 Member Function Documentation

6.125.3.1 virtual bool decaf::security::cert::Certificate::equals (const Certificate & *cert*) const [pure virtual]

Compares the encoded form of the two certificates.

Parameters:

cert *(p. 110)* The certificate to be tested for equality with this certificate.

Returns:

true if the given certificate is equal to this certificate.

6.125.3.2 virtual void decaf::security::cert::Certificate::getEncoded (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Provides the encoded form of this certificate.

Parameters:

output Receives the encoded form of this certificate.

Exceptions:

CertificateEncodingException *(p. 791)* if an encoding error occurs

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.125.3.3 virtual const PublicKey* decaf::security::cert::Certificate::getPublicKey () const [pure virtual]

Gets the public key of this certificate.

Returns:

the public key

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.125.3.4 virtual PublicKey* decaf::security::cert::Certificate::getPublicKey () [pure virtual]

Gets the public key of this certificate.

Returns:

the public key

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.125.3.5 virtual std::string decaf::security::cert::Certificate::getType () const [pure virtual]

Returns the type of this certificate.

Returns:

the type of this certificate

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

6.125.3.6 virtual std::string decaf::security::cert::Certificate::toString () const [pure virtual]

Returns a string representation of this certificate.

Returns:

a string representation of this certificate

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

6.125.3.7 virtual void decaf::security::cert::Certificate::verify (const PublicKey & *publicKey*, const std::string & *sigProvider*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key. Uses the verification engine of the specified provider.

Parameters:

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions:

NoSuchAlgorithmException *(*p. ***1942****)* - on unsupported signature algorithms.

InvalidKeyException *(*p. ***1467****)* - on incorrect key.

NoSuchProviderException *(*p. ***1948****)* - if there's no default provider.

SignatureException *(*p. ***2364****)* - on signature errors.

CertificateException *(*p. ***793****)* - on encoding errors.

6.125.3.8 virtual void decaf::security::cert::Certificate::verify (const PublicKey & *publicKey*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters:

publicKey The public key used to carry out the validation.

Exceptions:

NoSuchAlgorithmException (p. ***1942***) - on unsupported signature algorithms.

InvalidKeyException (p. ***1467***) - on incorrect key.

NoSuchProviderException (p. ***1948***) - if there's no default provider.

SignatureException (p. ***2364***) - on signature errors.

CertificateException (p. ***793***) - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**Certificate.h**

6.126 decaf::security::cert::CertificateEncodingException Class Reference

`#include <src/main/decaf/security/cert/CertificateEncodingException.h>`Inheritance diagram for decaf::security::cert::CertificateEncodingException:

Public Member Functions

- **CertificateEncodingException** () throw ()
  Default Constructor.
- **CertificateEncodingException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **CertificateEncodingException** (const **CertificateEncodingException** &ex) throw ()
  Copy Constructor.
- **CertificateEncodingException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **CertificateEncodingException** * **clone** () const
  Clones this exception.
- virtual ~**CertificateEncodingException** () throw ()

6.126.1 Constructor & Destructor Documentation

6.126.1.1 decaf::security::cert::CertificateEncodingException::CertificateEncodingException () throw () `[inline]`

Default Constructor.

6.126.1.2 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.126.1.3 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const CertificateEncodingException & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.126.1.4 decaf::security::cert::CertificateEncodingException::CertificateEncodingException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.126.1.5 virtual decaf::security::cert::CertificateEncodingException::~CertificateEncodingException () throw () [inline, virtual]

6.126.2 Member Function Documentation

6.126.2.1 virtual CertificateEncodingException* decaf::security::cert::CertificateEncodingException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 794).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateEncodingException.h**

6.127 decaf::security::cert::CertificateException Class Reference

`#include <src/main/decaf/security/cert/CertificateException.h>`Inheritance diagram for decaf::security::cert::CertificateException:

Public Member Functions

- **CertificateException** () throw ()

  Default Constructor.

- **CertificateException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **CertificateException** (const **CertificateException** &ex) throw ()

  Copy Constructor.

- **CertificateException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occured.

- virtual **CertificateException** * **clone** () const

  Clones this exception.

- virtual ~**CertificateException** () throw ()

6.127.1 Constructor & Destructor Documentation

6.127.1.1 decaf::security::cert::CertificateException::CertificateException () throw () [inline]

Default Constructor.

6.127.1.2 decaf::security::cert::CertificateException::CertificateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.127.1.3 decaf::security::cert::CertificateException::CertificateException (const CertificateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.127.1.4 decaf::security::cert::CertificateException::CertificateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.127.1.5 virtual decaf::security::cert::CertificateException::~CertificateException () throw () [inline, virtual]

6.127.2 Member Function Documentation

6.127.2.1 virtual CertificateException* decaf::security::cert::CertificateException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1381).

Reimplemented in **decaf::security::cert::CertificateEncodingException** (p. 792), **decaf::security::cert::CertificateExpiredException** (p. 796), **decaf::security::cert::CertificateNotYetValidException** (p. 798), and **decaf::security::cert::CertificateParsingException** (p. 800).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateException.h**

6.128 decaf::security::cert::CertificateExpiredException Class Reference

`#include <src/main/decaf/security/cert/CertificateExpiredException.h>`Inheritance diagram for decaf::security::cert::CertificateExpiredException:

Public Member Functions

- **CertificateExpiredException** () throw ()

  Default Constructor.

- **CertificateExpiredException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **CertificateExpiredException** (const **CertificateExpiredException** &ex) throw ()

  Copy Constructor.

- **CertificateExpiredException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occured.

- virtual **CertificateExpiredException** * **clone** () const

  Clones this exception.

- virtual ~**CertificateExpiredException** () throw ()

6.128.1 Constructor & Destructor Documentation

6.128.1.1 decaf::security::cert::CertificateExpiredException::CertificateExpiredException () throw () [inline]

Default Constructor.

6.128.1.2 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.128.1.3 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const CertificateExpiredException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.128.1.4 decaf::security::cert::CertificateExpiredException::CertificateExpiredException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.128.1.5 virtual decaf::security::cert::CertificateExpiredException::~CertificateExpiredException () throw () [inline, virtual]

6.128.2 Member Function Documentation

6.128.2.1 virtual CertificateExpiredException* decaf::security::cert::CertificateExpiredException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 794).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateExpiredException.h**

6.129 decaf::security::cert::CertificateNotYetValidException Class Reference

`#include <src/main/decaf/security/cert/CertificateNotYetValidException.h>`Inheritance diagram for decaf::security::cert::CertificateNotYetValidException:

Public Member Functions

- **CertificateNotYetValidException** () throw ()
  Default Constructor.
- **CertificateNotYetValidException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **CertificateNotYetValidException** (const **CertificateNotYetValidException** &ex) throw ()
  Copy Constructor.
- **CertificateNotYetValidException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **CertificateNotYetValidException** * **clone** () const
  Clones this exception.
- virtual ~**CertificateNotYetValidException** () throw ()

6.129.1 Constructor & Destructor Documentation

6.129.1.1 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException () throw () [inline]

Default Constructor.

6.129.1.2 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.129.1.3 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const CertificateNotYetValidException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.129.1.4 decaf::security::cert::CertificateNotYetValidException::CertificateNotYetValidException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.129.1.5 virtual decaf::security::cert::CertificateNotYetValidException::~CertificateNotYetValidException () throw () [inline, virtual]

6.129.2 Member Function Documentation

6.129.2.1 virtual CertificateNotYetValidException* decaf::security::cert::CertificateNotYetValidException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 794).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateNotYetValidException.h**

6.130 decaf::security::cert::CertificateParsingException Class Reference

`#include <src/main/decaf/security/cert/CertificateParsingException.h>`Inheritance diagram for decaf::security::cert::CertificateParsingException:

Public Member Functions

- **CertificateParsingException** () throw ()

  Default Constructor.

- **CertificateParsingException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **CertificateParsingException** (const **CertificateParsingException** &ex) throw ()

  Copy Constructor.

- **CertificateParsingException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occured.

- virtual **CertificateParsingException** * **clone** () const

  Clones this exception.

- virtual ~**CertificateParsingException** () throw ()

6.130.1 Constructor & Destructor Documentation

6.130.1.1 decaf::security::cert::CertificateParsingException::CertificateParsingException () throw () `[inline]`

Default Constructor.

6.130.1.2 decaf::security::cert::CertificateParsingException::CertificateParsingException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.130.1.3 decaf::security::cert::CertificateParsingException::CertificateParsingException (const CertificateParsingException & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.130.1.4 decaf::security::cert::CertificateParsingException::CertificateParsingException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.130.1.5 virtual decaf::security::cert::CertificateParsingException::~CertificateParsingException () throw () [inline, virtual]

6.130.2 Member Function Documentation

6.130.2.1 virtual CertificateParsingException* decaf::security::cert::CertificateParsingException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::cert::CertificateException** (p. 794).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**CertificateParsingException.h**

6.131 decaf::lang::Character Class Reference

`#include <src/main/decaf/lang/Character.h>`Inheritance diagram for decaf::lang::Character:

Public Member Functions

- **Character** (char value)
- virtual int **compareTo** (const **Character** &c) const

  *Compares this **Character** (p. 801) instance with another.*

- virtual bool **operator==** (const **Character** &c) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Character** &c) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const char &c) const

  *Compares this **Character** (p. 801) instance with a char type.*

- virtual bool **operator==** (const char &c) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const char &c) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- bool **equals** (const **Character** &c) const
- bool **equals** (const char &c) const
- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

Static Public Member Functions

- static **Character valueOf** (char value)

  *Returns a **Character** (p. 801) instance representing the specified char value.*

- static bool **isWhitespace** (char c)

  Indicates whether or not the given character is considered whitespace.

- static bool **isDigit** (char c)

  Indicates whether or not the given character is a digit.

- static bool **isLowerCase** (char c)

  Indicates whether or not the given character is a lower case character.

- static bool **isUpperCase** (char c)

  Indicates whether or not the given character is a upper case character.

- static bool **isLetter** (char c)

  Indicates whether or not the given character is a letter.

- static bool **isLetterOrDigit** (char c)

  Indicates whether or not the given character is either a letter or a digit.

- static bool **isISOControl** (char c)

  Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

- static int **digit** (char c, int radix)

  Returns the numeric value of the character ch in the specified radix.

Static Public Attributes

- static const int **MIN_RADIX** = 2

  The minimum radix available for conversion to and from strings.

- static const int **MAX_RADIX** = 36

  The maximum radix available for conversion to and from strings.

- static const char **MIN_VALUE** = (char)0x7F

  The minimum value that a signed char can take on.

- static const char **MAX_VALUE** = (char)0x80

  The maximum value that a signed char can take on.

- static const int **SIZE** = 8

  The size of the primitive charactor in bits.

6.131.1 Constructor & Destructor Documentation

6.131.1.1 decaf::lang::Character::Character (char *value*)

Parameters:

value - char to wrap.

6.131.2 Member Function Documentation

6.131.2.1 virtual unsigned char decaf::lang::Character::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.131.2.2 virtual int decaf::lang::Character::compareTo (const char & *c*) const [inline, virtual]

Compares this **Character** (p. 801) instance with a char type.

Parameters:

c - the char instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< char >** (p. 899).

6.131.2.3 virtual int decaf::lang::Character::compareTo (const Character & *c*) const [inline, virtual]

Compares this **Character** (p. 801) instance with another.

Parameters:

c - the **Character** (p. 801) instance to be compared

Returns:

zero if this object represents the same char value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

6.131.2.4 static int decaf::lang::Character::digit (char *c*, int *radix*) [static]

Returns the numeric value of the character ch in the specified radix. If the radix is not in the range MIN_RADIX <= radix <= MAX_RADIX or if the value of ch is not a valid digit in the specified radix, -1 is returned. A character is a valid digit if at least one of the following is true:

* The method isDigit is true of the character and the single-character decomposition is less than the specified radix. In this case the decimal digit value is returned. * The character is one of the uppercase Latin letters 'A' through 'Z' and its code is less than radix + 'A' - 10. In this case, ch - 'A' + 10 is returned. * The character is one of the lowercase Latin letters 'a' through 'z' and its code is less than radix + 'a' - 10. In this case, ch - 'a' + 10 is returned.

Parameters:

c - the char to be converted

radix - the radix of the number

Returns:

the numeric value of the number represented in the given radix

6.131.2.5 virtual double decaf::lang::Character::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.131.2.6 bool decaf::lang::Character::equals (const char & *c*) const [inline, virtual]

Returns:

true if the two Characters have the same value.

Implements **decaf::lang::Comparable< char >** (p. 900).

6.131.2.7 bool decaf::lang::Character::equals (const Character & *c*) const [inline]

Returns:

true if the two **Character** (p. 801) Objects have the same value.

6.131.2.8 virtual float decaf::lang::Character::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.131.2.9 virtual int decaf::lang::Character::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.131.2.10 static bool decaf::lang::Character::isDigit (char *c*) [inline, static]

Indicates whether or not the given character is a digit.

6.131.2.11 static bool decaf::lang::Character::isISOControl (char *c*) [inline, static]

Answers whether the character is an ISO control character, which is a char that lays in the range of 0 to 1f and 7f to 9f.

Parameters:

c - the character, including supplementary characters

Returns:

true if the char is an ISO control character

6.131.2.12 static bool decaf::lang::Character::isLetter (char *c*) [inline, static]

Indicates whether or not the given character is a letter.

6.131.2.13 static bool decaf::lang::Character::isLetterOrDigit (char *c*) [inline, static]

Indicates whether or not the given character is either a letter or a digit.

6.131.2.14 static bool decaf::lang::Character::isLowerCase (char *c*) [inline, static]

Indicates whether or not the given character is a lower case character.

6.131.2.15 static bool decaf::lang::Character::isUpperCase (char *c*) [inline, static]

Indicates whether or not the given character is a upper case character.

6.131.2.16 static bool decaf::lang::Character::isWhitespace (char *c*) [inline, static]

Indicates whether or not the given character is considered whitespace.

6.131.2.17 virtual long long decaf::lang::Character::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.131.2.18 virtual bool decaf::lang::Character::operator< (const char & c) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 900).

6.131.2.19 virtual bool decaf::lang::Character::operator< (const Character & c) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.131.2.20 virtual bool decaf::lang::Character::operator== (const char & c) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< char >** (p. 901).

6.131.2.21 virtual bool decaf::lang::Character::operator== (const Character & *c*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

c - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.131.2.22 virtual short decaf::lang::Character::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.131.2.23 std::string decaf::lang::Character::toString () const

Returns:

this **Character** (p. 801) Object as a String Representation

6.131.2.24 static Character decaf::lang::Character::valueOf (char *value*) [inline, static]

Returns a **Character** (p. 801) instance representing the specified char value.

Parameters:

value - the primitive char to wrap.

Returns:

a new Charactor instance that wraps this value.

6.131.3 Field Documentation

6.131.3.1 const int decaf::lang::Character::MAX_RADIX = 36 [static]

The maximum radix available for conversion to and from strings.

6.131.3.2 const char decaf::lang::Character::MAX_VALUE = (char)0x80 [static]

The maximum value that a signed char can take on.

6.131.3.3 const int decaf::lang::Character::MIN_RADIX = 2 [static]

The minimum radix available for conversion to and from strings.

6.131.3.4 const char decaf::lang::Character::MIN_VALUE = (char)0x7F [static]

The minimum value that a signed char can take on.

6.131.3.5 const int decaf::lang::Character::SIZE = 8 [static]

The size of the primitive charactor in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Character.h**

6.132 decaf::internal::nio::CharArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/CharArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::CharArrayBuffer:

Public Member Functions

- **CharArrayBuffer** (std::size_t capacity, bool **readOnly**=false)

  *Creates a **CharArrayBuffer** (p. 809) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **CharArrayBuffer** (char *array, std::size_t **offset**, std::size_t capacity, bool **readOnly**=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **CharArrayBuffer** (p. 809) object that wraps the given array.*

- **CharArrayBuffer** (**ByteArrayPerspective** &array, std::size_t **offset**, std::size_t length, bool **readOnly**=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **CharArrayBuffer** (const **CharArrayBuffer** &other)

  *Create a **CharArrayBuffer** (p. 809) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**CharArrayBuffer** ()
- virtual char * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the character array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual CharBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only char buffer that shares this buffer's content.

- virtual CharBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual CharBuffer * **duplicate** ()

  Creates a new char buffer that shares this buffer's content.

- virtual char **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual char **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible char array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual CharBuffer & **put** (char value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given char into this buffer at the current position, and then increments the position.

- virtual CharBuffer & **put** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given char into this buffer at the given index.

- virtual CharBuffer * **slice** () const

  Creates a new CharBuffer whose content is a shared subsequence of this buffer's content.

- virtual **lang::CharSequence** * **subSequence** (std::size_t start, std::size_t end) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

Protected Attributes

- bool **readOnly**
- **internal::nio::ByteArrayPerspective** * **_array**
- std::size_t **offset**

6.132.1 Constructor & Destructor Documentation

6.132.1.1 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **CharArrayBuffer** (p. 809) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.132.1.2 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (char * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = false) throw (decaf::lang::exceptions::NullPointerException)

Creates a **CharArrayBuffer** (p. 809) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.132.1.3 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *length*, bool *readOnly* = false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **CharArrayBuffer** (p. 809) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.132.1.4 decaf::internal::nio::CharArrayBuffer::CharArrayBuffer (const CharArrayBuffer & *other*)

Create a **CharArrayBuffer** (p. 809) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **CharArrayBuffer** (p. 809) this one is to mirror.

6.132.1.5 virtual decaf::internal::nio::CharArrayBuffer::~CharArrayBuffer () [virtual]

6.132.2 Member Function Documentation

6.132.2.1 virtual char* decaf::internal::nio::CharArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the character array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 822).

6.132.2.2 virtual std::size_t decaf::internal::nio::CharArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::CharBuffer** (p. 822).

6.132.2.3 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only char buffer which the caller then owns.

Implements **decaf::nio::CharBuffer** (p. 822).

6.132.2.4 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this CharBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 823).

6.132.2.5 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::duplicate () [virtual]

Creates a new char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new char Buffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 824).

6.132.2.6 virtual char decaf::internal::nio::CharArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the char at the given index.

Parameters:

index - the index in the Buffer where the char is to be read

Returns:

the char that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::CharBuffer** (p. 825).

6.132.2.7 virtual char decaf::internal::nio::CharArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the character at this buffer's current position, and then increments the position.

Returns:

the char at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::CharBuffer** (p. 825).

6.132.2.8 virtual bool decaf::internal::nio::CharArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible char array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::CharBuffer** (p. 826).

6.132.2.9 virtual bool decaf::internal::nio::CharArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.132.2.10 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (std::size_t *index*, char *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given char into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the char to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 828).

6.132.2.11 virtual CharBuffer& decaf::internal::nio::CharArrayBuffer::put (char *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters:

value - the char value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::CharBuffer** (p. 828).

6.132.2.12 virtual void decaf::internal::nio::CharArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.132.2.13 virtual CharBuffer* decaf::internal::nio::CharArrayBuffer::slice () const [virtual]

Creates a new CharBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create CharBuffer which the caller owns.

Implements **decaf::nio::CharBuffer** (p. 830).

6.132.2.14 virtual lang::CharSequence* decaf::internal::nio::CharArrayBuffer::subSequence (std::size_t *start*, std::size_t *end*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position. The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 631) + start, and its limit will be **position()** (p. 631) + end. The new Buffer will be read-only if, and only if, this buffer is read-only.

Parameters:

start - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 632)

end - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 632)

Returns:

The new character buffer, caller owns

Exceptions:

IndexOutOfBoundsException - If the preconditions on start and end fail

Implements **decaf::nio::CharBuffer** (p. 831).

6.132.3 Field Documentation

6.132.3.1 internal::nio::ByteArrayPerspective* decaf::internal::nio::CharArrayBuffer::_array [protected]

6.132.3.2 std::size_t decaf::internal::nio::CharArrayBuffer::offset [protected]

6.132.3.3 bool decaf::internal::nio::CharArrayBuffer::readOnly [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**CharArrayBuffer.h**

6.133 decaf::nio::CharBuffer Class Reference

This class defines four categories of operations upon character buffers:.

`#include <src/main/decaf/nio/CharBuffer.h>`Inheritance diagram for decaf::nio::CharBuffer:

Public Member Functions

- virtual ~**CharBuffer** ()
- virtual std::string **toString** () const
- **CharBuffer** & **append** (char value) throw (BufferOverflowException, ReadOnlyBufferException)

  Appends the specified character to this buffer.

- **CharBuffer** & **append** (const **lang::CharSequence** *value) throw (BufferOverflowException, ReadOnlyBufferException)

  Appends the specified character sequence to this buffer.

- **CharBuffer** & **append** (const **lang::CharSequence** *value, std::size_t start, std::size_t end) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException)

  Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

- virtual char * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the character array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **CharBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only char buffer that shares this buffer's content.

- char **charAt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Reads the character at the given index relative to the current position.

- virtual **CharBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **CharBuffer** * **duplicate** ()=0

  Creates a new char buffer that shares this buffer's content.

- virtual char **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual char **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **CharBuffer** & **get** (std::vector< char > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **CharBuffer** & **get** (char *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible char array.

- std::size_t **length** () const

  Returns the length of this character buffer.

- **CharBuffer** & **put** (**CharBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the chars remaining in the given source buffer into this buffer.

- **CharBuffer** & **put** (const char *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers chars into this buffer from the given source array.

- **CharBuffer** & **put** (std::vector< char > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source char array into this buffer.

- virtual **CharBuffer** & **put** (char value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given char into this buffer at the current position, and then increments the position.

- virtual **CharBuffer** & **put** (std::size_t index, char value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given char into this buffer at the given index.

- **CharBuffer** & **put** (const std::string &src, std::size_t start, std::size_t end) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)

  Relative bulk put method (optional operation).

- **CharBuffer** & **put** (const std::string &src) throw (BufferOverflowException, ReadOnlyBufferException)

  Relative bulk put method (optional operation).

- virtual std::size_t **read** (**CharBuffer** *target) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException)

  Attempts to read characters into the specified character buffer.

- virtual **lang::CharSequence** * **subSequence** (std::size_t start, std::size_t end) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position.

- virtual **CharBuffer** * **slice** () const =0

  *Creates a new **CharBuffer** (p. 817) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **CharBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **CharBuffer** &value) const
- virtual bool **operator==** (const **CharBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **CharBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **CharBuffer** * **allocate** (std::size_t capacity)

  Allocates a new character buffer.

- static **CharBuffer** * **wrap** (char *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **CharBuffer** (p. 817).*

- static **CharBuffer** * **wrap** (std::vector< char > &buffer)

  *Wraps the passed STL char Vector in a **CharBuffer** (p. 817).*

Protected Member Functions

- **CharBuffer** (std::size_t capacity)

  *Creates a **CharBuffer** (p. 817) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.133.1 Detailed Description

This class defines four categories of operations upon character buffers:. o Absolute and relative get and put methods that read and write single characters; o Relative bulk get methods that transfer contiguous sequences of characters from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of characters from a character array, a string, or some other character buffer into this buffer. o Methods for compacting, duplicating, and slicing a character buffer.

Character buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing character array or string into a buffer, or by creating a view of an existing byte buffer

This class implements the CharSequence interface so that character buffers may be used wherever character sequences are accepted, for example in the regular-expression package decaf.util.regex.

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained. The sequence of statements

cb.put("text/"); cb.put(subtype); cb.put("; charset="); cb.put(enc);

can, for example, be replaced by the single statement

cb.put("text/").put(subtype).put("; charset=").put(enc);

6.133.2 Constructor & Destructor Documentation

6.133.2.1 decaf::nio::CharBuffer::CharBuffer (std::size_t *capacity*) [protected]

Creates a **CharBuffer** (p. 817) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

6.133.2.2 virtual decaf::nio::CharBuffer::~CharBuffer () [inline, virtual]

6.133.3 Member Function Documentation

6.133.3.1 static CharBuffer* decaf::nio::CharBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new character buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Char buffer in chars (1 byte).

Returns:

the **CharBuffer** (p. 817) that was allocated, caller owns.

6.133.3.2 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*, std::size_t *start*, std::size_t *end*) throw (decaf::lang::exceptions::IndexOutOfBoundsException, BufferOverflowException, ReadOnlyBufferException) [virtual]

Appends a subsequence of the specified character sequence to this buffer If value is Null the the string "null" is appended to the buffer.

Parameters:

value - the CharSequence to append.

start - the index to start appending from.

end - the index to append to.

Returns:

a reference to this modified **CharBuffer** (p. 817)

Exceptions:

BufferOverflowException *(p. 658)* if there is no more space

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

IndexOutOfBoundsException if start > end, or > length of sequence.

Implements **decaf::lang::Appendable** (p. 484).

6.133.3.3 CharBuffer& decaf::nio::CharBuffer::append (const lang::CharSequence * *value*) throw (BufferOverflowException, ReadOnlyBufferException) [virtual]

Appends the specified character sequence to this buffer. If value is Null the the string "null" is appended to the buffer.

Parameters:

value - the CharSequence to append.

Returns:

a reference to this modified **CharBuffer** (p. 817)

Exceptions:

BufferOverflowException *(p. 658)* if there is no more space

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

Implements **decaf::lang::Appendable** (p. 485).

6.133.3.4 CharBuffer& decaf::nio::CharBuffer::append (char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [virtual]

Appends the specified character to this buffer.

Parameters:

value - the char to append.

Returns:

a reference to this modified **CharBuffer** (p. 817)

Exceptions:

BufferOverflowException *(*p. ***658)*** if there is no more space

ReadOnlyBufferException *(*p. ***2167)*** if this **Buffer** (p. 627) is read only.

Implements **decaf::lang::Appendable** (p. 485).

6.133.3.5 virtual char* decaf::nio::CharBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the character array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(*p. ***2167)*** if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 812).

6.133.3.6 virtual std::size_t decaf::nio::CharBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(*p. ***2167)*** if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 812).

6.133.3.7 virtual CharBuffer* decaf::nio::CharBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new

buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only char buffer which the caller then owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 812).

6.133.3.8 char decaf::nio::CharBuffer::charAt (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Reads the character at the given index relative to the current position.

Parameters:

index - The index of the character to be read relative to position

Returns:

The character at index **position()** (p. 631) + index

Exceptions:

IndexOutOfBoundsException

Implements **decaf::lang::CharSequence** (p. 833).

6.133.3.9 virtual CharBuffer& decaf::nio::CharBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **CharBuffer** (p. 817)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 813).

6.133.3.10 virtual int decaf::nio::CharBuffer::compareTo (const CharBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.133.3.11 virtual CharBuffer* decaf::nio::CharBuffer::duplicate () [pure virtual]

Creates a new char buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new char **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 813).

6.133.3.12 virtual bool decaf::nio::CharBuffer::equals (const CharBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.133.3.13 CharBuffer& decaf::nio::CharBuffer::get (char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers chars from this buffer into the given destination array. If there are fewer chars remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length chars from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length chars remaining in this buffer

NullPointerException if the passed buffer is null.

6.133.3.14 CharBuffer& decaf::nio::CharBuffer::get (std::vector< char > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers chars from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **CharBuffer** (p. 817)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length chars remaining in this buffer

6.133.3.15 virtual char decaf::nio::CharBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the char at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the char is to be read

Returns:

the char that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 813).

6.133.3.16 virtual char decaf::nio::CharBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the character at this buffer's current position, and then increments the position.

Returns:

the char at the current position

Exceptions:

BufferUnderflowException *(*p. ***661****)* if there no more data to return

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 814).

6.133.3.17 virtual bool decaf::nio::CharBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible char array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 814).

6.133.3.18 std::size_t decaf::nio::CharBuffer::length () const [inline, virtual]

Returns the length of this character buffer.

Returns:

the length of this buffer from the position to the limit.

Implements **decaf::lang::CharSequence** (p. 834).

6.133.3.19 virtual bool decaf::nio::CharBuffer::operator< (const CharBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.133.3.20 virtual bool decaf::nio::CharBuffer::operator== (const CharBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.133.3.21 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*) throw (BufferOverflowException, ReadOnlyBufferException)

Relative bulk put method (optional operation). This method transfers the entire content of the given source string into this buffer. An invocation of this method of the form dst.put(s) behaves in exactly the same way as the invocation

Parameters:

src - the string to copy from

Returns:

a reference to this **CharBuffer** (p. 817)

Exceptions:

BufferOverflowException *(p. **658**)* - If this buffer's current position is not

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

6.133.3.22 CharBuffer& decaf::nio::CharBuffer::put (const std::string & *src*, std::size_t *start*, std::size_t *end*) throw (BufferOverflowException, ReadOnlyBufferException, decaf::lang::exceptions::IndexOutOfBoundsException)

Relative bulk put method (optional operation). This method transfers characters from the given string into this buffer. If there are more characters to be copied from the string than remain in this buffer, that is, if end - start > **remaining()** (p. 632), then no characters are transferred and a **BufferOverflowException** (p. 658) is thrown.

Returns:

a reference to this buffer

Otherwise, this method copies n = end - start characters from the given string into this buffer, starting at the given start index and at the current position of this buffer. The position of this buffer is then incremented by n.

Parameters:

src - the string to copy from

start - position in src to start from

end - the position in src to stop at

Returns:

a reference to this **CharBuffer** (p. 817)

Exceptions:

BufferOverflowException *(p. **658**)* - If this buffer's current position is not

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.133.3.23 virtual CharBuffer& decaf::nio::CharBuffer::put (std::size_t *index*, char *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the char to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 814).

6.133.3.24 virtual CharBuffer& decaf::nio::CharBuffer::put (char *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given char into this buffer at the current position, and then increments the position.

Parameters:

value - the char value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 815).

6.133.3.25 CharBuffer& decaf::nio::CharBuffer::put (std::vector< char > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source char array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **CharBuffer** (p. 817)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. **658**)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

6.133.3.26 CharBuffer& decaf::nio::CharBuffer::put (const char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers chars into this buffer from the given source array. If there are more chars to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no chars are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which chars are to be read

offset- The offset within the array of the first char to be read;

length - The number of chars to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. **658**)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.133.3.27 CharBuffer& decaf::nio::CharBuffer::put (CharBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the chars remaining in the given source buffer into this buffer. If there are more chars remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 632), then no chars are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() chars from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take chars from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658)*** - If there is insufficient space in this buffer for the remaining chars in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(*p. ***2167)*** - If this buffer is read-only

6.133.3.28 virtual std::size_t decaf::nio::CharBuffer::read (CharBuffer * *target*) throw (decaf::lang::exceptions::NullPointerException, decaf::lang::exceptions::IllegalArgumentException, ReadOnlyBufferException) [virtual]

Attempts to read characters into the specified character buffer. The buffer is used as a repository of characters as-is: the only changes made are the results of a put operation. No flipping or rewinding of the buffer is performed.

Parameters:

target - the buffer to read characters into

Returns:

The number of characters added to the buffer, or string::npos if this source of characters is at its end

Exceptions:

NullPointerException - If target is Null

IllegalArgumentException - If target is this

ReadOnlyBufferException *(*p. ***2167)*** - If this buffer is read-only

6.133.3.29 virtual CharBuffer* decaf::nio::CharBuffer::slice () const [pure virtual]

Creates a new **CharBuffer** (p. 817) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **CharBuffer** (p. 817) which the caller owns.

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 815).

6.133.3.30 virtual lang::CharSequence* decaf::nio::CharBuffer::subSequence (std::size_t *start*, std::size_t *end*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Creates a new character buffer that represents the specified subsequence of this buffer, relative to the current position. The new buffer will share this buffer's content; that is, if the content of this buffer is mutable then modifications to one buffer will cause the other to be modified. The new buffer's capacity will be that of this buffer, its position will be **position()** (p. 631) + start, and its limit will be **position()** (p. 631) + end. The new **Buffer** (p. 627) will be read-only if, and only if, this buffer is read-only.

Parameters:

start - The index, relative to the current position, of the first character in the subsequence; must be non-negative and no larger than **remaining()** (p. 632)

end - The index, relative to the current position, of the character following the last character in the subsequence; must be no smaller than start and no larger than **remaining()** (p. 632)

Returns:

The new character buffer, caller owns

Exceptions:

IndexOutOfBoundsException - If the preconditions on start and end fail

Implements **decaf::lang::CharSequence** (p. 834).

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 816).

6.133.3.31 virtual std::string decaf::nio::CharBuffer::toString () const [virtual]

Returns:

a std::string describing this object

Implements **decaf::lang::CharSequence** (p. 834).

6.133.3.32 static CharBuffer* decaf::nio::CharBuffer::wrap (std::vector< char > & *buffer*) [static]

Wraps the passed STL char Vector in a **CharBuffer** (p. 817). The new buffer will be backed by the given char array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **CharBuffer** (p. 817) that is backed by buffer, caller owns.

6.133.3.33 static CharBuffer* decaf::nio::CharBuffer::wrap (char * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **CharBuffer** (p. 817). The new buffer will be backed by the given char array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **CharBuffer** (p. 817) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**CharBuffer.h**

6.134 decaf::lang::CharSequence Class Reference

A **CharSequence** (p. 833) is a readable sequence of char values.

`#include <src/main/decaf/lang/CharSequence.h>`Inheritance diagram for decaf::lang::CharSequence:

Public Member Functions

- virtual ~**CharSequence** ()
- virtual std::size_t **length** () const =0
- virtual char **charAt** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

- virtual **CharSequence** * **subSequence** (std::size_t start, std::size_t end) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  *Returns a new **CharSequence** (p. 833) that is a subsequence of this sequence.*

- virtual std::string **toString** () const =0

6.134.1 Detailed Description

A **CharSequence** (p. 833) is a readable sequence of char values. This interface provides uniform, read-only access to many different kinds of char sequences.

This interface does not define that a **CharSequence** (p. 833) should implement comparable, it is therefore up to the dervied classes that implement this interface to define equality, which implies that comparison of two CharSequences does not have a contract on equality.

6.134.2 Constructor & Destructor Documentation

6.134.2.1 virtual decaf::lang::CharSequence::~CharSequence () [inline, virtual]

6.134.3 Member Function Documentation

6.134.3.1 virtual char decaf::lang::CharSequence::charAt (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Returns the Char at the specified index so long as the index is not greater than the length of the sequence.

Parameters:

index - position to return the char at.

Returns:

the char at the given position

Exceptions:

IndexOutOfBoundsException if index is > than **length()** (p. 834)

Implemented in **decaf::nio::CharBuffer** (p. 823).

6.134.3.2 virtual std::size_t decaf::lang::CharSequence::length () const [pure virtual]

Returns:

the length of the underlying character sequence.

Implemented in **decaf::nio::CharBuffer** (p. 826).

6.134.3.3 virtual CharSequence* decaf::lang::CharSequence::subSequence (std::size_t *start*, std::size_t *end*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Returns a new **CharSequence** (p. 833) that is a subsequence of this sequence. The subsequence starts with the char value at the specified index and ends with the char value at index end - 1. The length (in chars) of the returned sequence is end - start, so if start == end then an empty sequence is returned.

Parameters:

start - the start index, inclusive

end - the end index, exclusive

Returns:

a new **CharSequence** (p. 833)

Exceptions:

IndexOutOfBoundsException if start or end > **length()** (p. 834)

Implemented in **decaf::internal::nio::CharArrayBuffer** (p. 816), and **decaf::nio::CharBuffer** (p. 831).

6.134.3.4 virtual std::string decaf::lang::CharSequence::toString () const [pure virtual]

Returns:

the string representation of this **CharSequence** (p. 833)

Implemented in **decaf::nio::CharBuffer** (p. 831).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**CharSequence.h**

6.135 decaf::lang::exceptions::ClassCastException Class Reference

`#include <src/main/decaf/lang/exceptions/ClassCastException.h>`Inheritance diagram for decaf::lang::exceptions::ClassCastException:

Public Member Functions

- **ClassCastException** () throw ()
  Default Constructor.
- **ClassCastException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **ClassCastException** (const **ClassCastException** &ex) throw ()
  Copy Constructor.
- **ClassCastException** (const std::exception *__cause__) throw ()
  Constructor.
- **ClassCastException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **ClassCastException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **ClassCastException** * **clone** () const
  Clones this exception.
- virtual ~**ClassCastException** () throw ()

6.135.1 Constructor & Destructor Documentation

6.135.1.1 decaf::lang::exceptions::ClassCastException::ClassCastException () throw () `[inline]`

Default Constructor.

6.135.1.2 decaf::lang::exceptions::ClassCastException::ClassCastException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.135.1.3 decaf::lang::exceptions::ClassCastException::ClassCastException (const ClassCastException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.135.1.4 decaf::lang::exceptions::ClassCastException::ClassCastException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.135.1.5 decaf::lang::exceptions::ClassCastException::ClassCastException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.135.1.6 decaf::lang::exceptions::ClassCastException::ClassCastException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.135.1.7 virtual decaf::lang::exceptions::ClassCastException::~ClassCastException () throw () [inline, virtual]

6.135.2 Member Function Documentation

6.135.2.1 virtual ClassCastException* decaf::lang::exceptions::ClassCastException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**ClassCastException.h**

6.136 cms::Closeable Class Reference

Interface for a class that implements the close method.

`#include <src/main/cms/Closeable.h>`Inheritance diagram for cms::Closeable:

Public Member Functions

- virtual ~**Closeable** ()
- virtual void **close** ()=0 throw (CMSException)

  Closes this object and deallocates the appropriate resources.

6.136.1 Detailed Description

Interface for a class that implements the close method. A class that implements this interface should release all resources upon the close call and should throw an exception from any methods that require those resources after they have been closed.

Since:

1.0

6.136.2 Constructor & Destructor Documentation

6.136.2.1 virtual cms::Closeable::~Closeable () `[inline, virtual]`

6.136.3 Member Function Documentation

6.136.3.1 virtual void cms::Closeable::close () throw (CMSException) `[pure virtual]`

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

CMSException *(*p. ***850)*** - If an error occurs while the resource is being closed.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 774), **activemq::cmsutil::CachedProducer** (p. 778), **activemq::cmsutil::PooledSession** (p. 2021), **activemq::commands::ActiveMQTempDestination** (p. 400), **activemq::core::ActiveMQConnection** (p. 193), **activemq::core::ActiveMQConsumer** (p. 222), **activemq::core::ActiveMQProducer** (p. 327), **activemq::core::ActiveMQSession** (p. 356), **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::failover::FailoverTransport** (p. 1299), **activemq::transport::IOTransport** (p. 1482), **activemq::transport::mock::MockTransport** (p. 1912), **activemq::transport::tcp::TcpTransport** (p. 2533), **activemq::transport::TransportFilter** (p. 2618), **activemq::wireformat::openwire::OpenWireFormatNegotia** (p. 1985), **cms::Connection** (p. 942), and **cms::Session** (p. 2273).

The documentation for this class was generated from the following file:

- src/main/cms/**Closeable.h**

6.137 decaf::io::Closeable Class Reference

Interface for a class that implements the close method.

`#include <src/main/decaf/io/Closeable.h>`Inheritance diagram for decaf::io::Closeable:

Public Member Functions

- virtual ~**Closeable** ()
- virtual void **close** ()=0 throw (lang::Exception)

  Closes this object and deallocates the appropriate resources.

6.137.1 Detailed Description

Interface for a class that implements the close method.

6.137.2 Constructor & Destructor Documentation

6.137.2.1 virtual decaf::io::Closeable::~Closeable () [inline, virtual]

6.137.3 Member Function Documentation

6.137.3.1 virtual void decaf::io::Closeable::close () throw (lang::Exception) [pure virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

CMSException

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 2400), **decaf::internal::io::StandardInputStream** (p. 2404), **decaf::internal::io::StandardOutputStream** (p. 2410), **decaf::io::BlockingByteArrayInputStream** (p. 568), **decaf::io::BufferedInputStream** (p. 635), **decaf::io::BufferedOutputStream** (p. 639), **decaf::io::ByteArrayInputStream** (p. 718), **decaf::io::ByteArrayOutputStream** (p. 724), **decaf::io::FilterInputStream** (p. 1317), **decaf::io::FilterOutputStream** (p. 1324), **decaf::net::BufferedSocket** (p. 642), **decaf::net::SocketInputStream** (p. 2385), **decaf::net::SocketOutputStream** (p. 2391), **decaf::net::TcpSocket** (p. 2526), and **decaf::util::logging::StreamHandler** (p. 2473).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Closeable.h**

6.138 activemq::transport::failover::CloseTransportsTask Class Reference

`#include <src/main/activemq/transport/failover/CloseTransportsTask.h>`Inheritance diagram for activemq::transport::failover::CloseTransportsTask:

Public Member Functions

- **CloseTransportsTask** ()
- virtual ~**CloseTransportsTask** ()
- void **add** (const **Pointer**< **Transport** > &transport)

  *Add a new **Transport** (p. 2608) to close.*

- virtual bool **isPending** () const

  This Task is pending if there are transports in the Queue that need to be closed.

- virtual bool **iterate** ()

  Return true until all transports have been closed and removed from the queue.

6.138.1 Constructor & Destructor Documentation

6.138.1.1 activemq::transport::failover::CloseTransportsTask::CloseTransportsTask () `[inline]`

6.138.1.2 virtual activemq::transport::failover::CloseTransportsTask::~CloseTransportsTask () `[inline, virtual]`

6.138.2 Member Function Documentation

6.138.2.1 void activemq::transport::failover::CloseTransportsTask::add (const Pointer< Transport > & *transport*)

Add a new **Transport** (p. 2608) to close.

6.138.2.2 virtual bool activemq::transport::failover::CloseTransportsTask::isPending () const `[virtual]`

This Task is pending if there are transports in the Queue that need to be closed.

Returns:

true if there is a **transport** (p. 67) in the queue that needs closed.

Implements **activemq::threads::CompositeTask** (p. 906).

6.138.2.3 virtual bool activemq::transport::failover::CloseTransportsTask::iterate () [virtual]

Return true until all transports have been closed and removed from the queue.

Implements **activemq::threads::Task** (p. 2520).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**CloseTransportsTask.h**

6.139 activemq::cmsutil::CmsAccessor Class Reference

Base class for **activemq.cmsutil.CmsTemplate** (p. 858) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 978) to operate on.

`#include <src/main/activemq/cmsutil/CmsAccessor.h>`Inheritance diagram for activemq::cmsutil::CmsAccessor:

Public Member Functions

- **CmsAccessor** ()
- virtual ~**CmsAccessor** ()
- virtual **ResourceLifecycleManager** * **getResourceLifecycleManager** ()
- virtual const **ResourceLifecycleManager** * **getResourceLifecycleManager** () const
- virtual void **setConnectionFactory** (**cms::ConnectionFactory** *connectionFactory)

  Set the ConnectionFactory to use for obtaining CMS Connections.

- virtual const **cms::ConnectionFactory** * **getConnectionFactory** () const

  Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

- virtual **cms::ConnectionFactory** * **getConnectionFactory** ()

  Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

- virtual void **setSessionAcknowledgeMode** (**cms::Session::AcknowledgeMode** sessionAcknowledgeMode)

  Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message.

- virtual **cms::Session::AcknowledgeMode** **getSessionAcknowledgeMode** () const

  Return the acknowledgment mode for CMS sessions.

Protected Member Functions

- virtual void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Initializes this object and prepares it for use.

- virtual void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Shuts down this object and destroys any allocated resources.

- virtual **cms::Connection** * **createConnection** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Create a CMS Connection via this template's ConnectionFactory.

- virtual **cms::Session** * **createSession** (**cms::Connection** *con) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

Create a CMS Session for the given Connection.

- virtual void **checkConnectionFactory** () throw (decaf::lang::exceptions::IllegalStateException)

  Verifies that the connection factory is valid.

6.139.1 Detailed Description

Base class for **activemq.cmsutil.CmsTemplate** (p. 858) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 978) to operate on. The subclass **activemq.cmsutil.CmsDestinationAccessor** (p. 847) adds further, destination-related properties.

Not intended to be used directly.

See also:

activemq.cmsutil.CmsDestinationAccessor (p. 847)
activemq.cmsutil.CmsTemplate (p. 858)

6.139.2 Constructor & Destructor Documentation

6.139.2.1 activemq::cmsutil::CmsAccessor::CmsAccessor ()

6.139.2.2 virtual activemq::cmsutil::CmsAccessor::~CmsAccessor () [virtual]

6.139.3 Member Function Documentation

6.139.3.1 virtual void activemq::cmsutil::CmsAccessor::checkConnectionFactory () throw (decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Verifies that the connection factory is valid.

6.139.3.2 virtual cms::Connection* activemq::cmsutil::CmsAccessor::createConnection () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Create a CMS Connection via this template's ConnectionFactory.

Returns:

the new CMS Connection

Exceptions:

cms::CMSException *(*p. ***850)*** if thrown by CMS API methods

6.139.3.3 virtual cms::Session* activemq::cmsutil::CmsAccessor::createSession (cms::Connection * *con*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Create a CMS Session for the given Connection.

Parameters:

con the CMS Connection to create a Session for

Returns:

the new CMS Session

Exceptions:

cms::CMSException (p. ***850***) if thrown by CMS API methods

6.139.3.4 virtual void activemq::cmsutil::CmsAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]

Shuts down this object and destroys any allocated resources.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 848), and **activemq::cmsutil::CmsTemplate** (p. 861).

6.139.3.5 virtual cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () [inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.139.3.6 virtual const cms::ConnectionFactory* activemq::cmsutil::CmsAccessor::getConnectionFactory () const [inline, virtual]

Return the ConnectionFactory that this accessor uses for obtaining CMS Connections.

6.139.3.7 virtual const ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () const [inline, virtual]

6.139.3.8 virtual ResourceLifecycleManager* activemq::cmsutil::CmsAccessor::getResourceLifecycleManager () [inline, virtual]

6.139.3.9 virtual cms::Session::AcknowledgeMode activemq::cmsutil::CmsAccessor::getSessionAcknowledgeMode () const [inline, virtual]

Return the acknowledgment mode for CMS sessions.

Returns:

the acknowledgment mode applied by this accessor

6.139.3.10 virtual void activemq::cmsutil::CmsAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [inline, protected, virtual]

Initializes this object and prepares it for use. This should be called before any other methods are called. This version does nothing.

Reimplemented in **activemq::cmsutil::CmsDestinationAccessor** (p. 848), and **activemq::cmsutil::CmsTemplate** (p. 864).

6.139.3.11 virtual void activemq::cmsutil::CmsAccessor::setConnectionFactory (cms::ConnectionFactory * *connectionFactory*) [inline, virtual]

Set the ConnectionFactory to use for obtaining CMS Connections.

6.139.3.12 virtual void activemq::cmsutil::CmsAccessor::setSessionAcknowledgeMode (cms::Session::AcknowledgeMode *sessionAcknowledgeMode*) [inline, virtual]

Set the CMS acknowledgment mode that is used when creating a CMS Session to send a message. Default is `AUTO_ACKNOWLEDGE`.

Parameters:

sessionAcknowledgeMode the acknowledgment mode

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsAccessor.h**

6.140 activemq::cmsutil::CmsDestinationAccessor Class Reference

Extends the `CmsAccessor` (p. 843) to add support for resolving destination names.

`#include <src/main/activemq/cmsutil/CmsDestinationAccessor.h>`Inheritance diagram for activemq::cmsutil::CmsDestinationAccessor:

Public Member Functions

- **CmsDestinationAccessor** ()
- virtual ~**CmsDestinationAccessor** ()
- virtual bool **isPubSubDomain** () const
- virtual void **setPubSubDomain** (bool pubSubDomain)
- virtual **DestinationResolver** * **getDestinationResolver** ()
- virtual const **DestinationResolver** * **getDestinationResolver** () const
- virtual void **setDestinationResolver** (**DestinationResolver** *destRes)

Protected Member Functions

- virtual void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Initializes the destination resolver.

- virtual void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Calls ***destroy()*** *(p. 848) on the destination resolver.*

- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Resolves the destination via the `DestinationResolver` *(p. 1211).*

- virtual void **checkDestinationResolver** () throw (decaf::lang::exceptions::IllegalStateException)

  Verifies that the destination resolver is valid.

6.140.1 Detailed Description

Extends the `CmsAccessor` (p. 843) to add support for resolving destination names. Not intended to be used directly.

See also:

CmsTemplate (p. 858)
CmsAccessor (p. 843)

6.140.2 Constructor & Destructor Documentation

6.140.2.1 activemq::cmsutil::CmsDestinationAccessor::CmsDestinationAccessor ()

6.140.2.2 virtual activemq::cmsutil::CmsDestinationAccessor::~CmsDestinationAccessor () [virtual]

6.140.3 Member Function Documentation

6.140.3.1 virtual void activemq::cmsutil::CmsDestinationAccessor::checkDestinationResolver () throw (decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Verifies that the destination resolver is valid.

6.140.3.2 virtual void activemq::cmsutil::CmsDestinationAccessor::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Calls **destroy()** (p. 848) on the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 845).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 861).

6.140.3.3 virtual const DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver () const [inline, virtual]

6.140.3.4 virtual DestinationResolver* activemq::cmsutil::CmsDestinationAccessor::getDestinationResolver () [inline, virtual]

6.140.3.5 virtual void activemq::cmsutil::CmsDestinationAccessor::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Initializes the destination resolver.

Reimplemented from **activemq::cmsutil::CmsAccessor** (p. 846).

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 864).

6.140.3.6 virtual bool activemq::cmsutil::CmsDestinationAccessor::isPubSubDomain () const [inline, virtual]

6.140.3.7 virtual cms::Destination* activemq::cmsutil::CmsDestinationAccessor::resolveDestinationName (cms::Session * *session*, const std::string & *destName*) throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Resolves the destination via the DestinationResolver (p. 1211).

Parameters:

session the session

destName the name of the destination.

Returns:

the destination

Exceptions:

cms::CMSException (p. ***850***) if resolution failed.

decaf::lang::exceptions::IllegalStateException (p. ***1400***) if the destination resolver property is NULL.

6.140.3.8 virtual void activemq::cmsutil::CmsDestinationAccessor::setDestinationResolver (DestinationResolver * *destRes*) [inline, virtual]

6.140.3.9 virtual void activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Reimplemented in **activemq::cmsutil::CmsTemplate** (p. 869).

Referenced by activemq::cmsutil::CmsTemplate::setPubSubDomain().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsDestinationAccessor.h**

6.141 cms::CMSException Class Reference

CMS API Exception that is the base for all exceptions thrown from CMS classes.

`#include <src/main/cms/CMSException.h>`Inheritance diagram for cms::CMSException:

Public Member Functions

- **CMSException** () throw ()
- **CMSException** (const **CMSException** &ex) throw ()
- **CMSException** (const std::string &message, const std::exception *cause) throw ()
- **CMSException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSException** () throw ()
- virtual std::string **getMessage** () const

  Gets the cause of the error.

- virtual const std::exception * **getCause** () const

  Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const

  Provides the stack trace for every point where this exception was caught, marked, and rethrown.

- virtual void **setMark** (const char *file, const int lineNumber)

  Adds a file/line number to the stack trace.

- virtual void **printStackTrace** () const

  Prints the stack trace to std::err.

- virtual void **printStackTrace** (std::ostream &stream) const

  Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const

  Gets the stack trace as one contiguous string.

6.141.1 Detailed Description

CMS API Exception that is the base for all exceptions thrown from CMS classes. This class represents an error that has occurred in CMS, providers can wrap provider specific exceptions in this class by setting the cause to an instance of a provider specific exception provided it can be cast to an std::exception.

Since the contained cause exception is of type std::exception and the C++ exception class has no clone or copy method defined the contained exception can only be owned by one instance of an **CMSException** (p. 850). To that end the class hands off the exception to each successive copy so care must be taken when handling **CMSException** (p. 850) instances.

Since:

1.0

6.141.2 Constructor & Destructor Documentation

6.141.2.1 cms::CMSException::CMSException () throw ()

6.141.2.2 cms::CMSException::CMSException (const CMSException & *ex*) throw ()

6.141.2.3 cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*) throw ()

6.141.2.4 cms::CMSException::CMSException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.141.2.5 virtual cms::CMSException::~CMSException () throw () [virtual]

6.141.3 Member Function Documentation

6.141.3.1 virtual const std::exception* cms::CMSException::getCause () const [virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of exceptions in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

6.141.3.2 virtual std::string cms::CMSException::getMessage () const [virtual]

Gets the cause of the error.

Returns:

string errors message

6.141.3.3 virtual std::vector< std::pair< std::string, int> > cms::CMSException::getStackTrace () const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns:

vector containing stack trace strings

6.141.3.4 virtual std::string cms::CMSException::getStackTraceString () const [virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

6.141.3.5 virtual void cms::CMSException::printStackTrace (std::ostream & *stream*) const [virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

6.141.3.6 virtual void cms::CMSException::printStackTrace () const [virtual]

Prints the stack trace to std::err.

6.141.3.7 virtual void cms::CMSException::setMark (const char * *file*, const int *lineNumber*) [virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use __FILE__).

lineNumber The line number in the calling file (use __LINE__).

The documentation for this class was generated from the following file:

- src/main/cms/**CMSException.h**

6.142 cms::CMSProperties Class Reference

Interface for a Java-like properties object.

`#include <src/main/cms/CMSProperties.h>`Inheritance diagram for cms::CMSProperties:

Public Member Functions

- virtual ~**CMSProperties** ()
- virtual bool **isEmpty** () const =0

  Returns true if the properties object is empty.

- virtual const char * **getProperty** (const std::string &name) const =0

  Looks up the value for the given property.

- virtual std::string **getProperty** (const std::string &name, const std::string &defaultValue) const =0

  Looks up the value for the given property.

- virtual void **setProperty** (const std::string &name, const std::string &value)=0

  Sets the value for a given property.

- virtual bool **hasProperty** (const std::string &name) const =0

  Check to see if the Property exists in the set.

- virtual void **remove** (const std::string &name)=0

  Removes the property with the given name.

- virtual std::vector< std::pair< std::string, std::string > > **toArray** () const =0

  Method that serializes the contents of the property map to an array.

- virtual void **copy** (const **CMSProperties** *source)=0

  Copies the contents of the given properties object to this one.

- virtual **CMSProperties** * **clone** () const =0

  Clones this object.

- virtual void **clear** ()=0

  Clears all properties from the map.

- virtual std::string **toString** () const =0

  Formats the contents of the Properties Object into a string that can be logged, etc.

6.142.1 Detailed Description

Interface for a Java-like properties object. This is essentially a map of key-value string pairs.

Since:

1.1

6.142.2 Constructor & Destructor Documentation

6.142.2.1 virtual cms::CMSProperties::~CMSProperties () [inline, virtual]

6.142.3 Member Function Documentation

6.142.3.1 virtual void cms::CMSProperties::clear () [pure virtual]

Clears all properties from the map.

Implemented in **activemq::util::ActiveMQProperties** (p. 333).

6.142.3.2 virtual CMSProperties* cms::CMSProperties::clone () const [pure virtual]

Clones this object.

Returns:

a replica of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 333).

6.142.3.3 virtual void cms::CMSProperties::copy (const CMSProperties * *source*) [pure virtual]

Copies the contents of the given properties object to this one.

Parameters:

source The source properties object.

6.142.3.4 virtual std::string cms::CMSProperties::getProperty (const std::string & *name*, const std::string & *defaultValue*) const [pure virtual]

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by `name`, if it exists, otherwise the `defaultValue`.

Implemented in **activemq::util::ActiveMQProperties** (p. 334).

6.142.3.5 virtual const char* cms::CMSProperties::getProperty (const std::string & *name*) const [pure virtual]

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Implemented in **activemq::util::ActiveMQProperties** (p. 334).

6.142.3.6 virtual bool cms::CMSProperties::hasProperty (const std::string & *name*) const [pure virtual]

Check to see if the Property exists in the set.

Parameters:

name the name of the property to check

Returns:

true if property exists, false otherwise.

Implemented in **activemq::util::ActiveMQProperties** (p. 334).

6.142.3.7 virtual bool cms::CMSProperties::isEmpty () const [pure virtual]

Returns true if the properties object is empty.

Returns:

true if empty

Implemented in **activemq::util::ActiveMQProperties** (p. 335).

6.142.3.8 virtual void cms::CMSProperties::remove (const std::string & *name*) [pure virtual]

Removes the property with the given name.

Parameters:

name the name of the property to be removed.s

Implemented in **activemq::util::ActiveMQProperties** (p. 335).

6.142.3.9 virtual void cms::CMSProperties::setProperty (const std::string & *name*, const std::string & *value*) [pure virtual]

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

Implemented in **activemq::util::ActiveMQProperties** (p. 335).

6.142.3.10 virtual std::vector< std::pair< std::string, std::string > > cms::CMSProperties::toArray () const [pure virtual]

Method that serializes the contents of the property map to an array.

Returns:

list of pairs where the first is the name and the second is the value.

Implemented in **activemq::util::ActiveMQProperties** (p. 335).

6.142.3.11 virtual std::string cms::CMSProperties::toString () const [pure virtual]

Formats the contents of the Properties Object into a string that can be logged, etc.

Returns:

string value of this object.

Implemented in **activemq::util::ActiveMQProperties** (p. 335).

The documentation for this class was generated from the following file:

- src/main/cms/**CMSProperties.h**

6.143 cms::CMSSecurityException Class Reference

This exception must be thrown when a provider rejects a user name/password submitted by a client.

`#include <src/main/cms/CMSSecurityException.h>`Inheritance diagram for cms::CMSSecurityException:

Public Member Functions

- **CMSSecurityException** () throw ()
- **CMSSecurityException** (const **CMSSecurityException** &ex) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause) throw ()
- **CMSSecurityException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**CMSSecurityException** () throw ()

6.143.1 Detailed Description

This exception must be thrown when a provider rejects a user name/password submitted by a client. It may also be thrown for any case where a security restriction prevents a method from completing.

Since:

1.3

6.143.2 Constructor & Destructor Documentation

6.143.2.1 cms::CMSSecurityException::CMSSecurityException () throw ()

6.143.2.2 cms::CMSSecurityException::CMSSecurityException (const CMSSecurityException & *ex*) throw ()

6.143.2.3 cms::CMSSecurityException::CMSSecurityException (const std::string & *message*, const std::exception * *cause*) throw ()

6.143.2.4 cms::CMSSecurityException::CMSSecurityException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.143.2.5 virtual cms::CMSSecurityException::~CMSSecurityException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**CMSSecurityException.h**

6.144 activemq::cmsutil::CmsTemplate Class Reference

`CmsTemplate` (p. 858) simplifies performing synchronous CMS operations.

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate:

Data Structures

- class **ProducerExecutor**
- class **ReceiveExecutor**
- class **ResolveProducerExecutor**
- class **ResolveReceiveExecutor**
- class **SendExecutor**

Public Member Functions

- **CmsTemplate** ()
- **CmsTemplate** (**cms::ConnectionFactory** *connectionFactory)
- virtual ~**CmsTemplate** ()
- virtual void **setDefaultDestination** (**cms::Destination** *defaultDestination)

  Sets the destination object to be used by default for send/receive operations.

- virtual const **cms::Destination** * **getDefaultDestination** () const

  Retrieves the default destination to be used for send/receive operations.

- virtual **cms::Destination** * **getDefaultDestination** ()

  Retrieves the default destination to be used for send/receive operations.

- virtual void **setDefaultDestinationName** (const std::string &defaultDestinationName)

  Sets the name of the default destination to be used from send/receive operations.

- virtual const std::string **getDefaultDestinationName** () const

  Gets the name of the default destination to be used for send/receive operations.

- virtual void **setPubSubDomain** (bool pubSubDomain)

  Indicates whether the default destination is a topic (true) or a queue (false).

- virtual void **setMessageIdEnabled** (bool messageIdEnabled)
- virtual bool **isMessageIdEnabled** () const
- virtual void **setMessageTimestampEnabled** (bool messageTimestampEnabled)
- virtual bool **isMessageTimestampEnabled** () const
- virtual void **setNoLocal** (bool noLocal)
- virtual bool **isNoLocal** () const
- virtual void **setReceiveTimeout** (long long receiveTimeout)
- virtual long long **getReceiveTimeout** () const
- virtual void **setExplicitQosEnabled** (bool explicitQosEnabled)

  Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

- virtual bool **isExplicitQosEnabled** () const

  If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message.

- virtual void **setDeliveryPersistent** (bool deliveryPersistent)

  Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false").

- virtual void **setDeliveryMode** (int deliveryMode)

  Set the delivery mode to use when sending a message.

- virtual int **getDeliveryMode** () const

  Return the delivery mode to use when sending a message.

- virtual void **setPriority** (int priority)

  Set the priority of a message when sending.

- virtual int **getPriority** () const

  Return the priority of a message when sending.

- virtual void **setTimeToLive** (long long timeToLive)

  Set the time-to-live of the message when sending.

- virtual long long **getTimeToLive** () const

  Return the time-to-live of the message when sending.

- virtual void **execute** (**SessionCallback** *action) throw (cms::CMSException)

  Executes the given action within a CMS Session.

- virtual void **execute** (**ProducerCallback** *action) throw (cms::CMSException)

  Executes the given action and provides it with a CMS Session and producer.

- virtual void **execute** (**cms::Destination** *dest, **ProducerCallback** *action) throw (cms::CMSException)

  Executes the given action and provides it with a CMS Session and producer.

- virtual void **execute** (const std::string &destinationName, **ProducerCallback** *action) throw (cms::CMSException)

  Executes the given action and provides it with a CMS Session and producer.

- virtual void **send** (**MessageCreator** *messageCreator) throw (cms::CMSException)

  Convenience method for sending a message to the default destination.

- virtual void **send** (**cms::Destination** *dest, **MessageCreator** *messageCreator) throw (cms::CMSException)

  Convenience method for sending a message to the specified destination.

- virtual void **send** (const std::string &destinationName, **MessageCreator** *messageCreator) throw (cms::CMSException)

Convenience method for sending a message to the specified destination.

- virtual **cms::Message** * **receive** () throw (cms::CMSException)

  Performs a synchronous read from the default destination.

- virtual **cms::Message** * **receive** (**cms::Destination** *destination) throw (cms::CMSException)

  Performs a synchronous read from the specified destination.

- virtual **cms::Message** * **receive** (const std::string &destinationName) throw (cms::CMSException)

  Performs a synchronous read from the specified destination.

- virtual **cms::Message** * **receiveSelected** (const std::string &selector) throw (cms::CMSException)

  Performs a synchronous read consuming only messages identified by the given selector.

- virtual **cms::Message** * **receiveSelected** (**cms::Destination** *destination, const std::string &selector) throw (cms::CMSException)

  Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

- virtual **cms::Message** * **receiveSelected** (const std::string &destinationName, const std::string &selector) throw (cms::CMSException)

  Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Static Public Attributes

- static const long long **RECEIVE_TIMEOUT_NO_WAIT** = -1

  Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

- static const long long **RECEIVE_TIMEOUT_INDEFINITE_WAIT** = 0

  Timeout value indicating a blocking receive without timeout.

- static const int **DEFAULT_PRIORITY** = 4

  Default message priority.

- static const long long **DEFAULT_TIME_TO_LIVE** = 0

  My default, messages should live forever.

Protected Member Functions

- void **init** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Initializes this object and prepares it for use.

- void **destroy** () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException)

  Clears all internal resources.

Friends

- class **ProducerExecutor**
- class **ResolveProducerExecutor**
- class **SendExecutor**
- class **ReceiveExecutor**
- class **ResolveReceiveExecutor**

6.144.1 Detailed Description

`CmsTemplate` (p. 858) simplifies performing synchronous CMS operations. This class is intended to be for CMS what Spring's `JmsTemplate` is for JMS. Provided with a CMS `ConnectionFactory`, creates and manages all other CMS resources internally.

Before using `CmsTemplate` (p. 858) the user must first set the destination (either by name or by setting the destination object directly) and then call `init` to initialize the object for use.

`CmsTemplate` (p. 858) allows the user to get access to a CMS `Session` through a user-defined `SessionCallback` (p. 2283). Similarly, if the user wants direct access to a CMS `MessageProducer`, it can provide a `ProducerCallback` (p. 2102). As a convenience, the user can bypass having to provide callbacks altogether for sending messages, by calling one of the `send` methods.

See also:

SessionCallback (p. 2283)
ProducerCallback (p. 2102)
MessageCreator (p. 1799)

6.144.2 Constructor & Destructor Documentation

6.144.2.1 activemq::cmsutil::CmsTemplate::CmsTemplate ()

6.144.2.2 activemq::cmsutil::CmsTemplate::CmsTemplate (cms::ConnectionFactory * *connectionFactory*)

6.144.2.3 virtual activemq::cmsutil::CmsTemplate::~CmsTemplate () `[virtual]`

6.144.3 Member Function Documentation

6.144.3.1 void activemq::cmsutil::CmsTemplate::destroy () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) `[protected, virtual]`

Clears all internal resources.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 848).

6.144.3.2 virtual void activemq::cmsutil::CmsTemplate::execute (const std::string & *destinationName*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

destinationName the name of the destination to send messages to (to internally be resolved to an actual destination)

action the action to perform

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs.

6.144.3.3 virtual void activemq::cmsutil::CmsTemplate::execute (cms::Destination * *dest*, ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

dest the destination to send messages to

action the action to perform

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs.

6.144.3.4 virtual void activemq::cmsutil::CmsTemplate::execute (ProducerCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action and provides it with a CMS Session and producer.

Parameters:

action the action to perform

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs.

6.144.3.5 virtual void activemq::cmsutil::CmsTemplate::execute (SessionCallback * *action*) throw (cms::CMSException) [virtual]

Executes the given action within a CMS Session.

Parameters:

action the action to perform within a CMS Session

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs.

6.144.3.6 virtual cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () [inline, virtual]

Retrieves the default destination to be used for send/receive operations.

Returns:

the default destination. Non-const version of this method.

6.144.3.7 virtual const cms::Destination* activemq::cmsutil::CmsTemplate::getDefaultDestination () const [inline, virtual]

Retrieves the default destination to be used for send/receive operations.

Returns:

the default destination. Const version of this method.

6.144.3.8 virtual const std::string activemq::cmsutil::CmsTemplate::getDefaultDestinationName () const [inline, virtual]

Gets the name of the default destination to be used for send/receive operations. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Returns:

the default name of the destination for send/receive operations.

6.144.3.9 virtual int activemq::cmsutil::CmsTemplate::getDeliveryMode () const [inline, virtual]

Return the delivery mode to use when sending a message.

6.144.3.10 virtual int activemq::cmsutil::CmsTemplate::getPriority () const [inline, virtual]

Return the priority of a message when sending.

6.144.3.11 virtual long long activemq::cmsutil::CmsTemplate::getReceiveTimeout () const [inline, virtual]

6.144.3.12 virtual long long activemq::cmsutil::CmsTemplate::getTimeToLive () const [inline, virtual]

Return the time-to-live of the message when sending.

6.144.3.13 void activemq::cmsutil::CmsTemplate::init () throw (cms::CMSException, decaf::lang::exceptions::IllegalStateException) [protected, virtual]

Initializes this object and prepares it for use. This should be called before any other methds are called.

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 848).

6.144.3.14 virtual bool activemq::cmsutil::CmsTemplate::isExplicitQosEnabled () const [inline, virtual]

If "true", then the values of deliveryMode, priority, and timeToLive will be used when sending a message. Otherwise, the default values, that may be set administratively, will be used.

Returns:

true if overriding default values of QOS parameters (deliveryMode, priority, and timeToLive)

See also:

setDeliveryMode (p. 868)
setPriority (p. 869)
setTimeToLive (p. 869)

6.144.3.15 virtual bool activemq::cmsutil::CmsTemplate::isMessageIdEnabled () const [inline, virtual]

6.144.3.16 virtual bool activemq::cmsutil::CmsTemplate::isMessageTimestampEnabled () const [inline, virtual]

6.144.3.17 virtual bool activemq::cmsutil::CmsTemplate::isNoLocal () const [inline, virtual]

6.144.3.18 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (const std::string & *destinationName*) throw (cms::CMSException) [virtual]

Performs a synchronous read from the specified destination.

Parameters:

destinationName the name of the destination to receive on (will be resolved to destination internally).

Returns:

the message

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs

6.144.3.19 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive (cms::Destination * *destination*) throw (cms::CMSException) [virtual]

Performs a synchronous read from the specified destination.

Parameters:

destination the destination to receive on

Returns:

the message

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs

6.144.3.20 virtual cms::Message* activemq::cmsutil::CmsTemplate::receive () throw (cms::CMSException) [virtual]

Performs a synchronous read from the default destination.

Returns:

the message

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs

6.144.3.21 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & *destinationName*, const std::string & *selector*) throw (cms::CMSException) [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters:

destinationName the name of the destination to receive on (will be resolved to destination internally).

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs

6.144.3.22 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (cms::Destination * *destination*, const std::string & *selector*) throw (cms::CMSException) [virtual]

Performs a synchronous read from the specified destination, consuming only messages identified by the given selector.

Parameters:

destination the destination to receive on.

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs

6.144.3.23 virtual cms::Message* activemq::cmsutil::CmsTemplate::receiveSelected (const std::string & *selector*) throw (cms::CMSException) [virtual]

Performs a synchronous read consuming only messages identified by the given selector.

Parameters:

selector the selector expression.

Returns:

the message

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs

6.144.3.24 virtual void activemq::cmsutil::CmsTemplate::send (const std::string & *destinationName*, MessageCreator * *messageCreator*) throw (cms::CMSException) [virtual]

Convenience method for sending a message to the specified destination.

Parameters:

destinationName The name of the destination to send to.

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs.

6.144.3.25 virtual void activemq::cmsutil::CmsTemplate::send (cms::Destination * *dest*, MessageCreator * *messageCreator*) throw (cms::CMSException) [virtual]

Convenience method for sending a message to the specified destination.

Parameters:

dest The destination to send to

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs.

6.144.3.26 virtual void activemq::cmsutil::CmsTemplate::send (MessageCreator * *messageCreator*) throw (cms::CMSException) [virtual]

Convenience method for sending a message to the default destination.

Parameters:

messageCreator Responsible for creating the message to be sent

Exceptions:

cms::CMSException *(*p. ***850)*** thrown if an error occurs.

6.144.3.27 virtual void activemq::cmsutil::CmsTemplate::setDefaultDestination (cms::Destination * *defaultDestination*) [inline, virtual]

Sets the destination object to be used by default for send/receive operations. If no default destination is provided, the `defaultDestinationName` property is used to resolve this default destination for send/receive operations.

Parameters:

defaultDestination the default destination

6.144.3.28 virtual void activemq::cmsutil::CmsTemplate::setDefaultDestinationName (const std::string & *defaultDestinationName*) [inline, virtual]

Sets the name of the default destination to be used from send/receive operations. Calling this method will set the `defaultDestination` property to NULL. The destination type (topic/queue) is determined by the `pubSubDomain` property.

Parameters:

defaultDestinationName the name of the destination for send/receive to by default.

6.144.3.29 virtual void activemq::cmsutil::CmsTemplate::setDeliveryMode (int *deliveryMode*) [inline, virtual]

Set the delivery mode to use when sending a message. Default is the Message default: "PERSISTENT".

Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters:

deliveryMode the delivery mode to use

See also:

isExplicitQosEnabled (p. 864)

6.144.3.30 virtual void activemq::cmsutil::CmsTemplate::setDeliveryPersistent (bool *deliveryPersistent*) [inline, virtual]

Set whether message delivery should be persistent or non-persistent, specified as boolean value ("true" or "false"). This will set the delivery mode accordingly, to either "PERSISTENT" or "NON_PERSISTENT".

Default it "true" aka delivery mode "PERSISTENT".

See also:

setDeliveryMode(int) (p. 868)

6.144.3.31 virtual void activemq::cmsutil::CmsTemplate::setExplicitQosEnabled (bool *explicitQosEnabled*) [inline, virtual]

Set if the QOS values (deliveryMode, priority, timeToLive) should be used for sending a message.

See also:

setDeliveryMode (p. 868)
setPriority (p. 869)
setTimeToLive (p. 869)

6.144.3.32 virtual void activemq::cmsutil::CmsTemplate::setMessageIdEnabled (bool *messageIdEnabled*) [inline, virtual]

6.144.3.33 virtual void activemq::cmsutil::CmsTemplate::setMessageTimestampEnabled (bool *messageTimestampEnabled*) [inline, virtual]

6.144.3.34 virtual void activemq::cmsutil::CmsTemplate::setNoLocal (bool *noLocal*) [inline, virtual]

6.144.3.35 virtual void activemq::cmsutil::CmsTemplate::setPriority (int *priority*) [inline, virtual]

Set the priority of a message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

See also:

isExplicitQosEnabled (p. 864)

6.144.3.36 virtual void activemq::cmsutil::CmsTemplate::setPubSubDomain (bool *pubSubDomain*) [inline, virtual]

Indicates whether the default destination is a topic (true) or a queue (false). Calling this method will set the `defaultDestination` property to NULL.

Parameters:

pubSubDomain indicates whether to use pub-sub messaging (topics).

Reimplemented from **activemq::cmsutil::CmsDestinationAccessor** (p. 849).

References activemq::cmsutil::CmsDestinationAccessor::setPubSubDomain().

6.144.3.37 virtual void activemq::cmsutil::CmsTemplate::setReceiveTimeout (long long *receiveTimeout*) [inline, virtual]

6.144.3.38 virtual void activemq::cmsutil::CmsTemplate::setTimeToLive (long long *timeToLive*) [inline, virtual]

Set the time-to-live of the message when sending. Since a default value may be defined administratively, this is only used when "isExplicitQosEnabled" equals "true".

Parameters:

timeToLive the message's lifetime (in milliseconds)

See also:

isExplicitQosEnabled (p. 864)

6.144.4 Friends And Related Function Documentation

6.144.4.1 friend class ProducerExecutor [friend]

6.144.4.2 friend class ReceiveExecutor [friend]

6.144.4.3 friend class ResolveProducerExecutor [friend]

6.144.4.4 friend class ResolveReceiveExecutor [friend]

6.144.4.5 friend class SendExecutor [friend]

6.144.5 Field Documentation

6.144.5.1 const int activemq::cmsutil::CmsTemplate::DEFAULT_PRIORITY = 4 [static]

Default message priority.

6.144.5.2 const long long activemq::cmsutil::CmsTemplate::DEFAULT_TIME_TO_LIVE = 0 [static]

My default, messages should live forever.

6.144.5.3 const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_INDEFINITE_WAIT = 0 [static]

Timeout value indicating a blocking receive without timeout.

6.144.5.4 const long long activemq::cmsutil::CmsTemplate::RECEIVE_TIMEOUT_NO_WAIT = -1 [static]

Timeout value indicating that a receive operation should check if a message is immediately available without blocking.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.145 decaf::util::Collection< E > Class Template Reference

The root interface in the collection hierarchy.

`#include <src/main/decaf/util/Collection.h>`Inheritance diagram for decaf::util::Collection< E >:

Public Member Functions

- virtual ~**Collection** ()
- virtual bool **add** (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

  Returns true if this collection changed as a result of the call.

- virtual bool **addAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

  Adds all of the elements in the specified collection to this collection.

- virtual void **clear** ()=0 throw (lang::exceptions::UnsupportedOperationException)

  Removes all of the elements from this collection (optional operation).

- virtual bool **contains** (const E &value) const =0 throw (lang::Exception)

  Returns true if this collection contains the specified element.

- virtual bool **containsAll** (const **Collection**< E > &collection) const =0 throw (lang::Exception)

  Returns true if this collection contains all of the elements in the specified collection.

- virtual bool **equals** (const **Collection**< E > &value) const =0

  Compares the passed collection to this one, if they contain the same elements, i.e.

- virtual bool **isEmpty** () const =0
- virtual bool **remove** (const E &value)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes a single instance of the specified element from the collection.

- virtual bool **removeAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes all this collection's elements that are also contained in the specified collection (optional operation).

- virtual bool **retainAll** (const **Collection**< E > &collection)=0 throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Retains only the elements in this collection that are contained in the specified collection (optional operation).

- virtual std::size_t **size** () const =0

  Returns the number of elements in this collection.

- virtual std::vector< E > **toArray** () const =0

  Returns an array containing all of the elements in this collection.

6.145.1 Detailed Description

template<typename E> class decaf::util::Collection< E >

The root interface in the collection hierarchy. A collection represents a group of objects, known as its elements. Some collections allow duplicate elements and others do not. Some are ordered and others unordered. This interface is typically used to pass collections around and manipulate them where maximum generality is desired.

All general-purpose **Collection** (p. 871) implementation classes (which typically implement **Collection** (p. 871) indirectly through one of its subinterfaces) should provide two "standard" constructors: a void (no arguments) constructor, which creates an empty collection, and a constructor with a single argument of type **Collection** (p. 871), which creates a new collection with the same elements as its argument. In effect, the latter constructor allows the user to copy any collection, producing an equivalent collection of the desired implementation type. There is no way to enforce this convention (as interfaces cannot contain constructors) but all of the general-purpose **Collection** (p. 871) implementations in the Decaf platform libraries comply.

The "destructive" methods contained in this interface, that is, the methods that modify the collection on which they operate, are specified to throw UnsupportedOperationException if this collection does not support the operation. If this is the case, these methods may, but are not required to, throw an UnsupportedOperationException if the invocation would have no effect on the collection. For example, invoking the addAll(Collection) method on an unmodifiable collection may, but is not required to, throw the exception if the collection to be added is empty.

Many methods in Collections Framework interfaces are defined in terms of the equals method. For example, the specification for the contains(Object o) method says: returns true if and only if this collection contains at least one element e such that (o==null ? e==null : o.equals(e)).

Since:

1.0

6.145.2 Constructor & Destructor Documentation

6.145.2.1 template<typename E> virtual decaf::util::Collection< E >::~Collection () [inline, virtual]

6.145.3 Member Function Documentation

6.145.3.1 template<typename E> virtual bool decaf::util::Collection< E >::add (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [pure virtual]

Returns true if this collection changed as a result of the call. (Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implemented in **decaf::util::AbstractQueue< E >** (p. 137), **decaf::util::StlList< E >** (p. 2421), **decaf::util::StlSet< E >** (p. 2448), **decaf::util::StlList< CompositeTask * >** (p. 2421), **decaf::util::StlList< URI >** (p. 2421), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2421), **decaf::util::StlList< PrimitiveValueNode >** (p. 2421), **decaf::util::StlList< Pointer< Command > >** (p. 2421), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2421), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2448), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2448).

6.145.3.2 template<typename E> virtual bool decaf::util::Collection< E >::addAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [pure virtual]

Adds all of the elements in the specified collection to this collection. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (This implies that the behavior of this call is undefined if the specified collection is this collection, and this collection is nonempty.)

Parameters:

collection - **Collection** (p. 871) whose elements are added to this one.

Returns:

true if this collection changed as a result of the call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

6.145.3.3 template<typename E> virtual void decaf::util::Collection< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [pure virtual]

Removes all of the elements from this collection (optional operation). This collection will be empty after this method returns unless it throws an exception.

Exceptions:

UnsupportedOperationException

Implemented in **decaf::util::AbstractCollection< E >** (p. 127), **decaf::util::AbstractQueue< E >** (p. 138), **decaf::util::StlList< E >** (p. 2423), **decaf::util::StlSet< E >** (p. 2449), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 127), **decaf::util::AbstractCollection< CompositeTask * >** (p. 127), **decaf::util::AbstractCollection< URI >** (p. 127), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 127), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 127), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 127), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 127), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 127), **decaf::util::StlList< CompositeTask * >** (p. 2423), **decaf::util::StlList< URI >** (p. 2423), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2423), **decaf::util::StlList< PrimitiveValueNode >** (p. 2423), **decaf::util::StlList< Pointer< Command > >** (p. 2423), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2423), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2449), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2449).

6.145.3.4 template<typename E> virtual bool decaf::util::Collection< E >::contains (const E & *value*) const throw (lang::Exception) [pure virtual]

Returns true if this collection contains the specified element. More formally, returns true if and only if this collection contains at least one element e such that (o==null ? e==null : o.equals(e)).

Parameters:

value - value to check for presence in the collection

Returns:

true if there is at least one of the elements in the collection

Exceptions:

Exception

Implemented in **decaf::util::AbstractCollection< E >** (p. 127), **decaf::util::StlList< E >** (p. 2423), **decaf::util::StlSet< E >** (p. 2450), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 127), **decaf::util::AbstractCollection< CompositeTask * >** (p. 127), **decaf::util::AbstractCollection< URI >** (p. 127), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 127), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 127), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 127), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 127), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 127), **decaf::util::StlList< CompositeTask * >** (p. 2423), **decaf::util::StlList< URI >** (p. 2423), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2423), **decaf::util::StlList< PrimitiveValueNode >** (p. 2423), **decaf::util::StlList< Pointer< Command > >** (p. 2423), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2423), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2450), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2450).

6.145.3.5 template<typename E> virtual bool decaf::util::Collection< E >::containsAll (const Collection< E > & *collection*) const throw (lang::Exception) [pure virtual]

Returns true if this collection contains all of the elements in the specified collection.

Parameters:

collection - **Collection** (p. 871) to compare to this one.

Exceptions:

Exception

6.145.3.6 template<typename E> virtual bool decaf::util::Collection< E >::equals (const Collection< E > & *value*) const [pure virtual]

Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.

Returns:

true if the Collections contain the same elements.

Implemented in **decaf::util::StlList< E >** (p. 2424), and **decaf::util::StlSet< E >** (p. 2450).

6.145.3.7 template<typename E> virtual bool decaf::util::Collection< E >::isEmpty () const [pure virtual]

Returns:

true if this collection contains no elements.

Implemented in **decaf::util::AbstractCollection< E >** (p. 129), **decaf::util::StlList< E >** (p. 2425), **decaf::util::StlSet< E >** (p. 2450), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 129), **decaf::util::AbstractCollection< CompositeTask * >** (p. 129), **decaf::util::AbstractCollection< URI >** (p. 129), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 129), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 129), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 129), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 129), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 129), **decaf::util::StlList< CompositeTask * >** (p. 2425), **decaf::util::StlList< URI >** (p. 2425), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2425), **decaf::util::StlList< PrimitiveValueNode >** (p. 2425), **decaf::util::StlList< Pointer< Command > >** (p. 2425), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2425), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2450), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2450).

6.145.3.8 template<typename E> virtual bool decaf::util::Collection< E >::remove (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]

Removes a single instance of the specified element from the collection. More formally, removes an element e such that (o==null ? e==null : o.equals(e)), if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

Parameters:

value - reference to the element to remove.

Returns:

true if the collection was changed

Exceptions:

UnsupportedOperationException

IllegalArgumentException

Implemented in **decaf::util::AbstractCollection< E >** (p. 130), **decaf::util::StlList< E >** (p. 2427), **decaf::util::StlSet< E >** (p. 2451), **decaf::util::AbstractCollection<**

Pointer< Synchronization > > (p. 130), **decaf::util::AbstractCollection< CompositeTask * >** (p. 130), **decaf::util::AbstractCollection< URI >** (p. 130), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 130), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 130), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 130), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 130), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 130), **decaf::util::StlList< CompositeTask * >** (p. 2427), **decaf::util::StlList< URI >** (p. 2427), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2427), **decaf::util::StlList< PrimitiveValueNode >** (p. 2427), **decaf::util::StlList< Pointer< Command > >** (p. 2427), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2427), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2451), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2451).

6.145.3.9 template<typename E> virtual bool decaf::util::Collection< E >::removeAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]

Removes all this collection's elements that are also contained in the specified collection (optional operation). After this call returns, this collection will contain no elements in common with the specified collection.

Parameters:

collection - The **Collection** (p. 871) whose elements are to be removed

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

6.145.3.10 template<typename E> virtual bool decaf::util::Collection< E >::retainAll (const Collection< E > & *collection*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [pure virtual]

Retains only the elements in this collection that are contained in the specified collection (optional operation). In other words, removes from this collection all of its elements that are not contained in the specified collection.

Parameters:

collection - The **Collection** (p. 871) whose elements are to be retained

Returns:

true if the collection changed as a result of this call

Exceptions:

UnsupportedOperationException

IllegalArgumentException

6.145.3.11 template<typename E> virtual std::size_t decaf::util::Collection< E >::size () const [pure virtual]

Returns the number of elements in this collection. If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns:

the number of elements in this collection

Implemented in **decaf::util::StlList< E >** (p. 2428), **decaf::util::StlSet< E >** (p. 2451), **decaf::util::StlList< CompositeTask * >** (p. 2428), **decaf::util::StlList< URI >** (p. 2428), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2428), **decaf::util::StlList< PrimitiveValueNode >** (p. 2428), **decaf::util::StlList< Pointer< Command > >** (p. 2428), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2428), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2451), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2451).

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::equals(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::isEmpty(), decaf::util::AbstractSet< ActiveMQSession * >::removeAll(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray().

6.145.3.12 template<typename E> virtual std::vector<E> decaf::util::Collection< E >::toArray () const [pure virtual]

Returns an array containing all of the elements in this collection. If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

This method acts as bridge between array-based and collection-based APIs.

Returns:

an array of the elements in this collection.

Implemented in **decaf::util::AbstractCollection< E >** (p. 132), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 132), **decaf::util::AbstractCollection< CompositeTask * >** (p. 132), **decaf::util::AbstractCollection< URI >** (p. 132), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 132), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 132), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 132), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 132), and **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 132).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Collection.h**

6.146 activemq::commands::Command Class Reference

`#include <src/main/activemq/commands/Command.h>`Inheritance diagram for activemq::commands::Command:

Public Member Functions

- virtual ~**Command** ()
- virtual void **setCommandId** (int id)=0

  *Sets the **Command** (p. 879) Id of this **Message** (p. 1737).*

- virtual int **getCommandId** () const =0

  *Gets the **Command** (p. 879) Id of this **Message** (p. 1737).*

- virtual void **setResponseRequired** (const bool required)=0

  *Set if this **Message** (p. 1737) requires a **Response** (p. 2231).*

- virtual bool **isResponseRequired** () const =0

  *Is a **Response** (p. 2231) required for this **Command** (p. 879).*

- virtual std::string **toString** () const =0

  Returns a provider-specific string that provides information about the contents of the command.

- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor)=0 throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

- virtual bool **isConnectionInfo** () const =0
- virtual bool **isConsumerInfo** () const =0
- virtual bool **isBrokerInfo** () const =0
- virtual bool **isKeepAliveInfo** () const =0
- virtual bool **isMessage** () const =0
- virtual bool **isMessageAck** () const =0
- virtual bool **isMessageDispatch** () const =0
- virtual bool **isMessageDispatchNotification** () const =0
- virtual bool **isProducerAck** () const =0
- virtual bool **isProducerInfo** () const =0
- virtual bool **isResponse** () const =0
- virtual bool **isRemoveInfo** () const =0
- virtual bool **isRemoveSubscriptionInfo** () const =0
- virtual bool **isShutdownInfo** () const =0
- virtual bool **isTransactionInfo** () const =0
- virtual bool **isWireFormatInfo** () const =0

6.146.1 Constructor & Destructor Documentation

6.146.1.1 virtual activemq::commands::Command::~Command () [inline, virtual]

6.146.2 Member Function Documentation

6.146.2.1 virtual int activemq::commands::Command::getCommandId () const [pure virtual]

Gets the **Command** (p. 879) Id of this **Message** (p. 1737).

Returns:

Command (p. 879) Id

Implemented in **activemq::commands::BaseCommand** (p. 509).

6.146.2.2 virtual bool activemq::commands::Command::isBrokerInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 509), and **activemq::commands::BrokerInfo** (p. 611).

6.146.2.3 virtual bool activemq::commands::Command::isConnectionInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::ConnectionInfo** (p. 1000).

6.146.2.4 virtual bool activemq::commands::Command::isConsumerInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::ConsumerInfo** (p. 1066).

6.146.2.5 virtual bool activemq::commands::Command::isKeepAliveInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::KeepAliveInfo** (p. 1557).

6.146.2.6 virtual bool activemq::commands::Command::isMessage () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::Message** (p. 1747).

6.146.2.7 virtual bool activemq::commands::Command::isMessageAck () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::MessageAck** (p. 1780).

6.146.2.8 virtual bool activemq::commands::Command::isMessageDispatch () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::MessageDispatch** (p. 1803).

6.146.2.9 virtual bool activemq::commands::Command::isMessageDispatchNotification () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 510), and **activemq::commands::MessageDispatchNotification** (p. 1826).

6.146.2.10 virtual bool activemq::commands::Command::isProducerAck () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::ProducerAck** (p. 2088).

6.146.2.11 virtual bool activemq::commands::Command::isProducerInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::ProducerInfo** (p. 2125).

6.146.2.12 virtual bool activemq::commands::Command::isRemoveInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::RemoveInfo** (p. 2179).

6.146.2.13 virtual bool activemq::commands::Command::isRemoveSubscriptionInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::RemoveSubscriptionInfo** (p. 2196).

6.146.2.14 virtual bool activemq::commands::Command::isResponse () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::Response** (p. 2233).

6.146.2.15 virtual bool activemq::commands::Command::isResponseRequired () const [pure virtual]

Is a **Response** (p. 2231) required for this **Command** (p. 879).

Returns:

true if a response is required.

Implemented in **activemq::commands::BaseCommand** (p. 511).

6.146.2.16 virtual bool activemq::commands::Command::isShutdownInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 511), and **activemq::commands::ShutdownInfo** (p. 2351).

6.146.2.17 virtual bool activemq::commands::Command::isTransactionInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 512), and **activemq::commands::TransactionInfo** (p. 2591).

6.146.2.18 virtual bool activemq::commands::Command::isWireFormatInfo () const [pure virtual]

Implemented in **activemq::commands::BaseCommand** (p. 512), and **activemq::commands::WireFormatInfo** (p. 2706).

6.146.2.19 virtual void activemq::commands::Command::setCommandId (int *id*) [pure virtual]

Sets the **Command** (p. 879) Id of this **Message** (p. 1737).

Parameters:

id **Command** (p. 879) Id

Implemented in **activemq::commands::BaseCommand** (p. 512).

6.146.2.20 virtual void activemq::commands::Command::setResponseRequired (const bool *required*) [pure virtual]

Set if this **Message** (p. 1737) requires a **Response** (p. 2231).

Parameters:

required true if response is required

Implemented in **activemq::commands::BaseCommand** (p. 512).

6.146.2.21 virtual std::string activemq::commands::Command::toString () const [pure virtual]

Returns a provider-specific string that provides information about the contents of the command.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 150), **activemq::commands::ActiveMQBytesMessage** (p. 173), **activemq::commands::ActiveMQMapMessage** (p. 266), **activemq::commands::ActiveMQMessage** (p. 281), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQStreamMessage** (p. 382), **activemq::commands::ActiveMQTextMessage** (p. 452), **activemq::commands::BaseCommand** (p. 512), **activemq::commands::BrokerInfo** (p. 612), **activemq::commands::ConnectionControl** (p. 948), **activemq::commands::ConnectionError** (p. 964), **activemq::commands::ConnectionInfo** (p. 1001), **activemq::commands::ConsumerControl** (p. 1031), **activemq::commands::ConsumerInfo** (p. 1067), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::DestinationInfo** (p. 1197), **activemq::commands::ExceptionResponse** (p. 1278), **activemq::commands::FlushCommand** (p. 1359), **activemq::commands::IntegerResponse** (p. 1444), **activemq::commands::KeepAliveInfo** (p. 1557), **activemq::commands::Message** (p. 1749), **activemq::commands::MessageAck** (p. 1781), **activemq::commands::MessageDispatch** (p. 1803), **activemq::commands::MessageDispatchNotification** (p. 1827), **activemq::commands::MessagePull** (p. 1896), **activemq::commands::ProducerAck** (p. 2088), **activemq::commands::ProducerInfo** (p. 2125), **activemq::commands::RemoveInfo** (p. 2179), **activemq::commands::RemoveSubscriptionInfo** (p. 2196), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2233), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2351), **activemq::commands::TransactionInfo** (p. 2592), and **activemq::commands::WireFormatInfo** (p. 2708).

6.146.2.22 virtual decaf::lang::Pointer<commands::Command> activemq::commands::Command::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [pure virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implemented in **activemq::commands::BrokerError** (p. 589), **activemq::commands::BrokerInfo** (p. 613), **activemq::commands::ConnectionControl** (p. 948), **activemq::commands::ConnectionError** (p. 965), **activemq::commands::ConnectionInfo** (p. 1001), **activemq::commands::ConsumerControl** (p. 1032), **activemq::commands::ConsumerInfo** (p. 1068), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DestinationInfo** (p. 1197), **activemq::commands::FlushCommand** (p. 1359), **activemq::commands::KeepAliveInfo** (p. 1557), **activemq::commands::Message** (p. 1750), **activemq::commands::MessageAck** (p. 1781), **activemq::commands::MessageDispatch**

(p. 1804), **activemq::commands::MessageDispatchNotification** (p. 1827), **activemq::commands::MessagePull** (p. 1896), **activemq::commands::ProducerAck** (p. 2089), **activemq::commands::ProducerInfo** (p. 2126), **activemq::commands::RemoveInfo** (p. 2179), **activemq::commands::RemoveSubscriptionInfo** (p. 2196), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2234), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2351), **activemq::commands::TransactionInfo** (p. 2592), and **activemq::commands::WireFormatInfo** (p. 2708).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Command.h**

6.147 activemq::state::CommandVisitor Class Reference

Interface for an Object that can visit the various Command Objects that are sent from and to this client.

`#include <src/main/activemq/state/CommandVisitor.h>`Inheritance diagram for activemq::state::CommandVisitor:

Public Member Functions

- virtual ~**CommandVisitor** ()
- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (**commands::ConnectionId** *id)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (**commands::SessionId** *id)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (**commands::ProducerId** *id)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (**commands::ConsumerId** *id)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (**commands::DestinationInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (**commands::DestinationInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (**commands::Message** *send)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (**commands::MessageAck** *ack)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (**commands::MessagePull** *pull)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (**commands::WireFormatInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (**commands::KeepAliveInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (**commands::ShutdownInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (**commands::FlushCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (**commands::BrokerInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processForgetTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processEndTransaction** (**commands::TransactionInfo** *info)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerAck** (**commands::ProducerAck** *ack)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (**commands::MessageDispatch** *dispatch)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processControlCommand** (**commands::ControlCommand** *command)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionError** (**commands::ConnectionError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionControl** (**commands::ConnectionControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerControl** (**commands::ConsumerControl** *control)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerError** (**commands::BrokerError** *error)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (**commands::ReplayCommand** *replay)=0 throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processResponse** (**commands::Response** *response)=0 throw (exceptions::ActiveMQException)

6.147.1 Detailed Description

Interface for an Object that can visit the various Command Objects that are sent from and to this client. The Commands themselves implement a `visit` method that is called with an instance of this interface and each one then call the appropriate `processXXX` method.

Since:

3.0

6.147.2 Constructor & Destructor Documentation

6.147.2.1 virtual activemq::state::CommandVisitor::~CommandVisitor () [inline, virtual]

6.147.3 Member Function Documentation

6.147.3.1 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBeginTransaction (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1024).

6.147.3.2 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerError (commands::BrokerError * *error*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.3 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processBrokerInfo (commands::BrokerInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.4 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionOnePhase (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1024).

6.147.3.5 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processCommitTransactionTwoPhase (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1024).

6.147.3.6 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionControl (commands::ConnectionControl * *control*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.7 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionError (commands::ConnectionError * *error*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.8 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConnectionInfo (commands::ConnectionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.9 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConsumerControl (commands::ConsumerControl * *control*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.10 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processConsumerInfo (commands::ConsumerInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.11 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processControlCommand (commands::ControlCommand * *command*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.12 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processDestinationInfo (commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.13 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processEndTransaction (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.14 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processFlushCommand (commands::FlushCommand * *command*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.15 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processForgetTransaction (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.16 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processKeepAliveInfo (commands::KeepAliveInfo * *info*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.17 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessage (commands::Message * *send*) throw (exceptions::ActiveMQException)** `[pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.18 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageAck (commands::MessageAck * *ack*) throw (exceptions::ActiveMQException)** `[pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.19 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageDispatch (commands::MessageDispatch * *dispatch*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.20 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessageDispatchNotification (commands::MessageDispatchNotification * *notification*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.21 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processMessagePull (commands::MessagePull * *pull*) throw (exceptions::ActiveMQException)** `[pure virtual]`

6.147.3.22 **virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processPrepareTransaction (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException)** `[pure virtual]`

Implemented in **activemq::state::ConnectionStateTracker** (p. 1025).

6.147.3.23 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processProducerAck (commands::ProducerAck * *ack*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.24 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processProducerInfo (commands::ProducerInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.25 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRecoverTransactions (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.26 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveConnection (commands::ConnectionId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.27 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveConsumer (commands::ConsumerId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.28 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveDestination (commands::DestinationInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.29 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveInfo (commands::RemoveInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::CommandVisitorAdapter** (p. 896).

6.147.3.30 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveProducer (commands::ProducerId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.31 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveSession (commands::SessionId * *id*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.32 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.33 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processReplayCommand (commands::ReplayCommand * *replay*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.34 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processResponse (commands::Response * *response*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.35 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processRollbackTransaction (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1026).

6.147.3.36 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processSessionInfo (commands::SessionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::ConnectionStateTracker** (p. 1027).

6.147.3.37 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processShutdownInfo (commands::ShutdownInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

6.147.3.38 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processTransactionInfo (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

Implemented in **activemq::state::CommandVisitorAdapter** (p. 897).

6.147.3.39 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitor::processWireFormat (commands::WireFormatInfo * *info*) throw (exceptions::ActiveMQException) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitor.h**

6.148 activemq::state::CommandVisitorAdapter Class Reference

Default Implementation of a **CommandVisitor** (p. 885) that returns NULL for all calls.

`#include <src/main/activemq/state/CommandVisitorAdapter.h>`Inheritance diagram for activemq::state::CommandVisitorAdapter:

Public Member Functions

- virtual ~**CommandVisitorAdapter** ()
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConnection** (**commands::ConnectionId** *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSession** (**commands::SessionId** *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveProducer** (**commands::ProducerId** *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveConsumer** (**commands::ConsumerId** *id AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processDestinationInfo** (**commands::DestinationInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveDestination** (**commands::DestinationInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRemoveSubscriptionInfo** (**commands::RemoveSubscriptionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessage** (**commands::Message** *send AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageAck** (**commands::MessageAck** *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessagePull** (**commands::MessagePull** *pull AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBeginTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processPrepareTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionOnePhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processCommitTransactionTwoPhase** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRollbackTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processWireFormat** (**commands::WireFormatInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processKeepAliveInfo** (**commands::KeepAliveInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processShutdownInfo** (**commands::ShutdownInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processFlushCommand** (**commands::FlushCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processBrokerInfo** (**commands::BrokerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processRecoverTransactions** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processForgetTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processEndTransaction** (**commands::TransactionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatchNotification** (**commands::MessageDispatchNotification** *notification AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processProducerAck** (**commands::ProducerAck** *ack AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processMessageDispatch** (**commands::MessageDispatch** *dispatch AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processControlCommand** (**commands::ControlCommand** *command AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionError** (**commands::ConnectionError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConnectionControl** (**commands::ConnectionControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)
- virtual **decaf::lang::Pointer< commands::Command > processConsumerControl** (**commands::ConsumerControl** *control AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processBrokerError** (**commands::BrokerError** *error AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processReplayCommand** (**commands::ReplayCommand** *replay AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processResponse** (**commands::Response** *response AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processConnectionInfo** (**commands::ConnectionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processSessionInfo** (**commands::SessionInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processProducerInfo** (**commands::ProducerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processConsumerInfo** (**commands::ConsumerInfo** *info AMQCPP_UNUSED) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processTransactionInfo** (**commands::TransactionInfo** *info) throw (exceptions::ActiveMQException)

- virtual **decaf::lang::Pointer< commands::Command > processRemoveInfo** (**commands::RemoveInfo** *info) throw (exceptions::ActiveMQException)

6.148.1 Detailed Description

Default Implementation of a **CommandVisitor** (p. 885) that returns NULL for all calls.

Since:

3.0

6.148.2 Constructor & Destructor Documentation

6.148.2.1 virtual activemq::state::CommandVisitorAdapter::~CommandVisitorAdapter () [inline, virtual]

6.148.3 Member Function Documentation

6.148.3.1 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBeginTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.2 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBrokerError (commands::BrokerError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.3 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processBrokerInfo (commands::BrokerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.4 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionOnePhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.5 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processCommitTransactionTwoPhase (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.6 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionControl (commands::ConnectionControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.7 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionError (commands::ConnectionError *error *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.8 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConnectionInfo (commands::ConnectionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.9 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConsumerControl (commands::ConsumerControl *control *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.10 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processConsumerInfo (commands::ConsumerInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.11 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processControlCommand (commands::ControlCommand *command *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

References activemq::commands::ConnectionId::ID_CONNECTIONID, activemq::commands::ConsumerId::ID_CONSUMERID, activemq::commands::ProducerId::ID_PRODUCERID, and activemq::commands::SessionId::ID_SESSIONID.

6.148.3.30 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveProducer (commands::ProducerId *id *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.31 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSession (commands::SessionId *id *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.32 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRemoveSubscriptionInfo (commands::RemoveSubscriptionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.33 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processReplayCommand (commands::ReplayCommand *replay *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.34 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processResponse (commands::Response *response *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.35 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processRollbackTransaction (commands::TransactionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.36 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processSessionInfo (commands::SessionInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.37 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processShutdownInfo (commands::ShutdownInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

6.148.3.38 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processTransactionInfo (commands::TransactionInfo * *info*) throw (exceptions::ActiveMQException) [inline, virtual]

Implements **activemq::state::CommandVisitor** (p. 891).

References activemq::core::ActiveMQConstants::TRANSACTION_STATE_BEGIN, activemq::core::ActiveMQConstants::TRANSACTION_STATE_-

COMMITONEPHASE, activemq::core::ActiveMQConstants::TRANSACTION_STATE_COMMITTWOPHASE, activemq::core::ActiveMQConstants::TRANSACTION_STATE_END, activemq::core::ActiveMQConstants::TRANSACTION_STATE_FORGET, activemq::core::ActiveMQConstants::TRANSACTION_STATE_PREPARE, activemq::core::ActiveMQConstants::TRANSACTION_STATE_RECOVER, and activemq::core::ActiveMQConstants::TRANSACTION_STATE_ROLLBACK.

6.148.3.39 virtual decaf::lang::Pointer<commands::Command> activemq::state::CommandVisitorAdapter::processWireFormat (commands::WireFormatInfo *info *AMQCPP_UNUSED*) throw (exceptions::ActiveMQException) [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**CommandVisitorAdapter.h**

6.149 decaf::lang::Comparable< T > Class Template Reference

This interface imposes a total ordering on the objects of each class that implements it.

`#include <src/main/decaf/lang/Comparable.h>`

Public Member Functions

- virtual ~**Comparable** ()
- virtual int **compareTo** (const T &value) const =0

  Compares this object with the specified object for order.

- virtual bool **equals** (const T &value) const =0
- virtual bool **operator==** (const T &value) const =0

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const T &value) const =0

  Compares this object to another and returns true if this object is considered to be less than the one passed.

6.149.1 Detailed Description

template<typename T> class decaf::lang::Comparable< T >

This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

6.149.2 Constructor & Destructor Documentation

6.149.2.1 template<typename T> virtual decaf::lang::Comparable< T >::~Comparable () [inline, virtual]

6.149.3 Member Function Documentation

6.149.3.1 template<typename T> virtual int decaf::lang::Comparable< T >::compareTo (const T & *value*) const [pure virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation sgn(expression) designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure sgn(x.compareTo(y)) == -sgn(y.compareTo(x)) for all x and y. (This implies that x.compareTo(y) must throw an exception iff y.compareTo(x) throws an exception.)

The implementor must also ensure that the relation is transitive: (x.compareTo(y)>0 && y.compareTo(z)>0) implies x.compareTo(z)>0.

Finally, the implementer must ensure that x.compareTo(y)==0 implies that sgn(x.compareTo(z)) == sgn(y.compareTo(z)), for all z.

It is strongly recommended, but not strictly required that (x.compareTo(y)==0) == (x.equals(y)). Generally speaking, any class that implements the **Comparable** (p. 899) interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Implemented in **decaf::lang::Boolean** (p. 574), **decaf::lang::Byte** (p. 666), **decaf::lang::Character** (p. 803), **decaf::lang::Double** (p. 1234), **decaf::lang::Float** (p. 1331), **decaf::lang::Integer** (p. 1431), **decaf::lang::Long** (p. 1655), and **decaf::lang::Short** (p. 2323).

6.149.3.2 template<typename T> virtual bool decaf::lang::Comparable< T >::equals (const T & *value*) const [pure virtual]

Returns:

true if this value is considered equal to the passed value.

Implemented in **decaf::lang::Boolean** (p. 575), **decaf::lang::Byte** (p. 667), **decaf::lang::Character** (p. 804), **decaf::lang::Double** (p. 1236), **decaf::lang::Float** (p. 1332), **decaf::lang::Integer** (p. 1432), **decaf::lang::Long** (p. 1656), and **decaf::lang::Short** (p. 2324).

6.149.3.3 template<typename T> virtual bool decaf::lang::Comparable< T >::operator< (const T & *value*) const [pure virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 575), **decaf::lang::Byte** (p. 668), **decaf::lang::Character** (p. 806), **decaf::lang::Double** (p. 1238), **decaf::lang::Float** (p. 1334), **decaf::lang::Integer** (p. 1435), **decaf::lang::Long** (p. 1659), and **decaf::lang::Short** (p. 2325).

6.149.3.4 template<typename T> virtual bool decaf::lang::Comparable< T >::operator== (const T & *value*) const [pure virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implemented in **decaf::lang::Boolean** (p. 576), **decaf::lang::Byte** (p. 669), **decaf::lang::Character** (p. 806), **decaf::lang::Double** (p. 1238), **decaf::lang::Float** (p. 1335), **decaf::lang::Integer** (p. 1435), **decaf::lang::Long** (p. 1659), and **decaf::lang::Short** (p. 2326).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Comparable.h**

6.150 decaf::util::Comparator< T > Class Template Reference

A comparison function, which imposes a total ordering on some collection of objects.

`#include <src/main/decaf/util/Comparator.h>`

Public Member Functions

- virtual ~**Comparator** ()
- virtual bool **operator()** (const T &left, const T &right) const =0

  *Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 902) to be passed to an STL **Map** (p. 1689) for use as the sorting criteria.*

- virtual int **compare** (const T &o1, const T &o2) const =0

  Compares its two arguments for order.

6.150.1 Detailed Description

template<typename T> class decaf::util::Comparator< T >

A comparison function, which imposes a total ordering on some collection of objects. Comparators can be passed to a sort method (such as Collections.sort) to allow precise control over the sort order. Comparators can also be used to control the order of certain data structures.

The ordering imposed by a **Comparator** (p. 902) c on a set of elements S is said to be consistent with equals if and only if (compare(e1, e2) == 0) has the same boolean value as (e1 == e2) for every e1 and e2 in S.

6.150.2 Constructor & Destructor Documentation

6.150.2.1 template<typename T> virtual decaf::util::Comparator< T >::~Comparator () [inline, virtual]

6.150.3 Member Function Documentation

6.150.3.1 template<typename T> virtual int decaf::util::Comparator< T >::compare (const T & *o1*, const T & *o2*) const [pure virtual]

Compares its two arguments for order. Returns a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

The implementor must ensure that sgn(compare(x, y)) == -sgn(compare(y, x)) for all x and y. (This implies that compare(x, y) must throw an exception if and only if compare(y, x) throws an exception.)

The implementor must also ensure that the relation is transitive: ((compare(x, y)>0) && (compare(y, z)>0)) implies compare(x, z)>0.

Finally, the implementer must ensure that compare(x, y)==0 implies that sgn(compare(x, z))==sgn(compare(y, z)) for all z.

It is generally the case, but not strictly required that (compare(x, y)==0) == (x == y)). Generally speaking, any comparator that violates this condition should clearly indicate this fact. The recommended language is "Note: this comparator imposes orderings that are inconsistent with equals."

Parameters:

o1 - the first object to be compared

o2 - the second object to be compared

Returns:

a negative integer, zero, or a positive integer as the first argument is less than, equal to, or greater than the second.

6.150.3.2 template<typename T> virtual bool decaf::util::Comparator< T >::operator() (const T & *left*, const T & *right*) const [pure virtual]

Implementation of the Binary function interface as a means of allowing a **Comparator** (p. 902) to be passed to an STL **Map** (p. 1689) for use as the sorting criteria.

Parameters:

left - the Left hand side operand.

right - the Right hand side operand.

Returns:

true if the vale of left is less than the value of right.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Comparator.h**

6.151 activemq::util::CompositeData Class Reference

Represents a Composite URI.

`#include <src/main/activemq/util/CompositeData.h>`

Public Member Functions

- **CompositeData** ()
- virtual ~**CompositeData** ()
- **StlList**< **URI** > & **getComponents** ()
- const **StlList**< **URI** > & **getComponents** () const
- void **setComponents** (const **StlList**< **URI** > &components)
- std::string **getFragment** () const
- void **setFragment** (const std::string &fragment)
- const **Properties** & **getParameters** () const
- void **setParameters** (const **Properties** &parameters)
- std::string **getScheme** () const
- void **setScheme** (const std::string &scheme)
- std::string **getPath** () const
- void **setPath** (const std::string &path)
- std::string **getHost** () const
- void **setHost** (const std::string &host)
- **URI toURI** () const throw (decaf::net::URISyntaxException)

6.151.1 Detailed Description

Represents a Composite URI.

Since:

3.0

6.151.2 Constructor & Destructor Documentation

6.151.2.1 activemq::util::CompositeData::CompositeData ()

6.151.2.2 virtual activemq::util::CompositeData::~CompositeData () [virtual]

6.151.3 Member Function Documentation

6.151.3.1 const StlList<URI>& activemq::util::CompositeData::getComponents () const [inline]

6.151.3.2 StlList<URI>& activemq::util::CompositeData::getComponents () [inline]

6.151.3.3 std::string activemq::util::CompositeData::getFragment () const [inline]

6.151.3.4 std::string activemq::util::CompositeData::getHost () const [inline]

6.151.3.5 const Properties& activemq::util::CompositeData::getParameters () const [inline]

6.151.3.6 std::string activemq::util::CompositeData::getPath () const [inline]

6.151.3.7 std::string activemq::util::CompositeData::getScheme () const [inline]

6.151.3.8 void activemq::util::CompositeData::setComponents (const StlList< URI > & *components*) [inline]

6.151.3.9 void activemq::util::CompositeData::setFragment (const std::string & *fragment*) [inline]

6.151.3.10 void activemq::util::CompositeData::setHost (const std::string & *host*) [inline]

6.151.3.11 void activemq::util::CompositeData::setParameters (const Properties & *parameters*) [inline]

6.151.3.12 void activemq::util::CompositeData::setPath (const std::string & *path*) [inline]

6.151.3.13 void activemq::util::CompositeData::setScheme (const std::string & *scheme*) [inline]

6.151.3.14 URI activemq::util::CompositeData::toURI () const throw (decaf::net::URISyntaxException)

The documentation for this class was generated from the following file:

- src/main/activemq/util/**CompositeData.h**

6.152 activemq::threads::CompositeTask Class Reference

Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p. 908).

`#include <src/main/activemq/threads/CompositeTask.h>`Inheritance diagram for activemq::threads::CompositeTask:

Public Member Functions

- virtual ~**CompositeTask** ()
- virtual bool **isPending** () const =0

  *Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p. 2520) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to **wakeup**.*

6.152.1 Detailed Description

Represents a single task that can be part of a set of Tasks that are contained in a `CompositeTaskRunner` (p. 908).

Since:

3.0

6.152.2 Constructor & Destructor Documentation

6.152.2.1 virtual activemq::threads::CompositeTask::~CompositeTask () [inline, virtual]

6.152.3 Member Function Documentation

6.152.3.1 virtual bool activemq::threads::CompositeTask::isPending () const [pure virtual]

Indicates whether this task has any pending work that needs to be done, if not then it is skipped and the next **Task** (p. 2520) in the CompositeTaskRunner's list of tasks is checked, if none of the tasks have any pending work to do, then the runner can go to sleep until it awakened by a call to `wakeup`.

Since:

3.0

Implemented in **activemq::transport::failover::BackupTransportPool** (p. 505), **activemq::transport::failover::CloseTransportsTask** (p. 841), and **activemq::transport::failover::FailoverTransport** (p. 1302).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTask.h**

6.153 activemq::threads::CompositeTaskRunner Class Reference

A **Task** (p. 2520) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

`#include <src/main/activemq/threads/CompositeTaskRunner.h>`Inheritance diagram for activemq::threads::CompositeTaskRunner:

Public Member Functions

- **CompositeTaskRunner** ()
- virtual ~**CompositeTaskRunner** ()
- void **addTask** (**CompositeTask** *task)

  *Adds a new **CompositeTask** (p. 906) to the Set of Tasks that this class manages.*

- void **removeTask** (**CompositeTask** *task)

  *Removes a **CompositeTask** (p. 906) that was added previously.*

- virtual void **shutdown** (unsigned int timeout)

  Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()

  Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()

  *Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()

  Run method - called by the Thread class in the context of the thread.

- virtual bool **iterate** ()

  Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.153.1 Detailed Description

A **Task** (p. 2520) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.

Since:

3.0

6.153.2 Constructor & Destructor Documentation

6.153.2.1 activemq::threads::CompositeTaskRunner::CompositeTaskRunner ()

6.153.2.2 virtual activemq::threads::CompositeTaskRunner::~CompositeTaskRunner () [virtual]

6.153.3 Member Function Documentation

6.153.3.1 void activemq::threads::CompositeTaskRunner::addTask (CompositeTask * *task*)

Adds a new **CompositeTask** (p. 906) to the Set of Tasks that this class manages.

Parameters:

task - Pointer to a **CompositeTask** (p. 906) instance.

6.153.3.2 virtual bool activemq::threads::CompositeTaskRunner::iterate () [protected, virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implements **activemq::threads::Task** (p. 2520).

6.153.3.3 void activemq::threads::CompositeTaskRunner::removeTask (CompositeTask * *task*)

Removes a **CompositeTask** (p. 906) that was added previously.

Parameters:

task - Pointer to a **CompositeTask** (p. 906) instance.

6.153.3.4 virtual void activemq::threads::CompositeTaskRunner::run () [protected, virtual]

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2256).

6.153.3.5 virtual void activemq::threads::CompositeTaskRunner::shutdown () [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2522).

6.153.3.6 virtual void activemq::threads::CompositeTaskRunner::shutdown (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 2522).

6.153.3.7 virtual void activemq::threads::CompositeTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2523).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**CompositeTaskRunner.h**

6.154 activemq::transport::CompositeTransport Class Reference

A Composite **Transport** (p. 2608) is a **Transport** (p. 2608) implementation that is composed of several Transports.

`#include <src/main/activemq/transport/CompositeTransport.h>`Inheritance diagram for activemq::transport::CompositeTransport:

Public Member Functions

- virtual ~**CompositeTransport** ()
- virtual void **addURI** (const **List**< **URI** > &uris)=0

  *Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2608) is a composite of.*

- virtual void **removeURI** (const **List**< **URI** > &uris)=0

  *Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2608) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2608) should result in that **Transport** (p. 2608) being disposed of.*

6.154.1 Detailed Description

A Composite **Transport** (p. 2608) is a **Transport** (p. 2608) implementation that is composed of several Transports. The composition could be such that only one **Transport** (p. 2608) exists for each URI that is composed or there could be many active Transports working at once.

Since:

3.0

6.154.2 Constructor & Destructor Documentation

6.154.2.1 virtual activemq::transport::CompositeTransport::~CompositeTransport () [inline, virtual]

6.154.3 Member Function Documentation

6.154.3.1 virtual void activemq::transport::CompositeTransport::addURI (const List< URI > & *uris*) [pure virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2608) is a composite of.

Parameters:

uris The new URI set to add to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1299).

6.154.3.2 virtual void activemq::transport::CompositeTransport::removeURI (const List< URI > & *uris*) [pure virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2608) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2608) should result in that **Transport** (p. 2608) being disposed of.

Parameters:

uris The new URI set to remove to the set this composite maintains.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1303).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**CompositeTransport.h**

6.155 decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR > Class Template Reference

Interface for a **Map** (p. 1689) type that provides additional **atomic** (p. 118) putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1689) interface.

`#include <src/main/decaf/util/concurrent/ConcurrentMap.h>`Inheritance diagram for decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >:

Public Member Functions

- virtual ~**ConcurrentMap** ()
- virtual bool **putIfAbsent** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  If the specified key is not already associated with a value, associate it with the given value.

- virtual bool **remove** (const K &key, const V &value)=0

  Remove entry for key only if currently mapped to given value.

- virtual bool **replace** (const K &key, const V &oldValue, const V &newValue)=0

  Replace entry for key only if currently mapped to given value.

- virtual V **replace** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)

  Replace entry for key only if currently mapped to some value.

6.155.1 Detailed Description

template<typename K, typename V, typename COMPARATOR> class decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >

Interface for a **Map** (p. 1689) type that provides additional **atomic** (p. 118) putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1689) interface.

Since:

1.0

6.155.2 Constructor & Destructor Documentation

6.155.2.1 template<typename K, typename V, typename COMPARATOR> virtual decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::~ConcurrentMap () [inline, virtual]

6.155.3 Member Function Documentation

6.155.3.1 template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::putIfAbsent (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key The key to map the value to.

value The value to map to the given key.

Returns:

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions:

UnsupportedOperationException if the put operation is not supported by this map

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 927), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 927).

6.155.3.2 template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::remove (const K & *key*, const V & *value*) [pure virtual]

Remove entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == value ) ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value associated with the specified key.

Returns:

true if the value was removed, false otherwise

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 928), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 928), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 928), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 928), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 928), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 928), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 928).

6.155.3.3 template<typename K, typename V, typename COMPARATOR> virtual V decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Replace entry for key only if currently mapped to some value. Acts as

```
if( ( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns:

copy of the previous value associated with specified key, or throws an NoSuchElementException if there was no mapping for key.

Exceptions:

NoSuchElementException if there was no previous mapping.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 929), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 929).

6.155.3.4 template<typename K, typename V, typename COMPARATOR> virtual bool decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >::replace (const K & *key*, const V & *oldValue*, const V & *newValue*) [pure virtual]

Replace entry for key only if currently mapped to given value. Acts as

```
if( ( map.containsKey( key ) && ( map.get( key ) == oldValue  ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns:

true if the value was replaced

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 930), and **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 930).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentMap.h**

6.156 decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

`#include <src/main/decaf/util/concurrent/ConcurrentStlMap.h>`Inheritance diagram for decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >:

Public Member Functions

- **ConcurrentStlMap** ()

  Default constructor - does nothing.

- **ConcurrentStlMap** (const **ConcurrentStlMap** &source)

  Copy constructor - copies the content of the given map into this one.

- **ConcurrentStlMap** (const **Map**< K, V, COMPARATOR > &source)

  Copy constructor - copies the content of the given map into this one.

- virtual ~**ConcurrentStlMap** ()
- virtual bool **equals** (const **ConcurrentStlMap** &source) const

  Comparison, equality is dependent on the method of determining if the element are equal.
  Parameters:

  source *- **Map** (p. 1689) to compare to this one.*
  Returns:

  *true if the **Map** (p. 1689) passed is equal in value to this one.*

- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const
- virtual void **copy** (const **ConcurrentStlMap** &source)

  Copies the content of the source map into this map.
  Erases all existing data in this map.
  Parameters:

  source *The source object to copy from.*

- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)
- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)

  Removes all keys and values from this map.
  Exceptions:

  UnsupportedOperationException *if this map is unmodifiable.*

- virtual bool **containsKey** (const K &key) const

  Indicates whether or this map contains a value for the given key.
  Parameters:

  key *The key to look up.*
  Returns:

  true if this map contains the value, otherwise false.

- virtual bool **containsValue** (const V &value) const

  Indicates whether or this map contains a value for the given value, i.e.
  they are equal, this is done by operator== so the types must pass equivalence testing in this manner.
  Parameters:

  value *The Value to look up.*
  Returns:

  true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const
  Returns:

  *if the **Map** (p. 1689) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const
  Returns:

  The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*
  If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.
  Parameters:

  key *The search key.*
  Returns:

  A reference to the value for the given key.
  Exceptions:

  NoSuchElementException *if the key requests doesn't exist in the **Map** (p. 1689).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*
  If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.
  Parameters:

  key *The search key.*
  Returns:

  A {const} reference to the value for the given key.
  Exceptions:

  NoSuchElementException *if the key requests doesn't exist in the **Map** (p. 1689).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

  Sets the value for the specified key.
  Parameters:

  key *The target key.*
  value *The value to be set.*
  Exceptions:

  UnsupportedOperationException *if this map is unmodifiable.*

- virtual void **putAll** (const **ConcurrentStlMap**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

  *Stores a copy of the Mappings contained in the other **Map** (p. 1689) in this one.*

  Parameters:

  other *A **Map** (p. 1689) instance whose elements are to all be inserted in this **Map** (p. 1689).*

  Exceptions:

  UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

  Parameters:

  key *The search key.*

  Returns:

  a copy of the element that was previously mapped to the given key

  Exceptions:

  NoSuchElementException *if this key is not in the **Map** (p. 1689).*
  UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

  *Returns a **Set** (p. 2320) view of the mappings contained in this map.*
  *The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1490), **Set.remove** (p. 130), removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

  Returns:

  the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

  Returns:

  the entire set of values in this map as a std::vector.

- bool **putIfAbsent** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

  If the specified key is not already associated with a value, associate it with the given value.

- bool **remove** (const K &key, const V &value)

  Remove entry for key only if currently mapped to given value.

- bool **replace** (const K &key, const V &oldValue, const V &newValue)

Replace entry for key only if currently mapped to given value.

- V **replace** (const K &key, const V &value) throw (decaf::lang::exceptions::NoSuchElementException)

  Replace entry for key only if currently mapped to some value.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.156.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map. This version of **Map** (p. 1689) extends the **ConcurrentMap** (p. 913) inteface and implements all the methods defined in that interface. Unlike a Java ConcurrentHashMap this implementations synchronizes all methods such that any call to this class will block if another thread is already holding a lock, much like the Java HashTable.

Since:

1.0

6.156.2 Constructor & Destructor Documentation

6.156.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap () [inline]

Default constructor - does nothing.

6.156.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap (const ConcurrentStlMap< K, V, COMPARATOR > & *source*) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.156.2.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::ConcurrentStlMap (const Map< K, V, COMPARATOR > & *source*) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.156.2.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::~ConcurrentStlMap () [inline, virtual]

6.156.3 Member Function Documentation

6.156.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1690).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.156.3.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::containsKey (const K & *key*) const [inline, virtual]

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1691).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::remove(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace().

6.156.3.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::containsValue (const V & *value*) const [inline, virtual]

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1692).

6.156.3.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & *source*) [inline, virtual]

6.156.3.5 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::copy (const ConcurrentStlMap< K, V, COMPARATOR > & *source*) [inline, virtual]

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::ConcurrentStlMap().

6.156.3.6 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & *source*) const [inline, virtual]

6.156.3.7 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::equals (const ConcurrentStlMap< K, V, COMPARATOR > & *source*) const [inline, virtual]

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - **Map** (p. 1689) to compare to this one.

Returns:

true if the **Map** (p. 1689) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

6.156.3.8 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & *key*) const throw (lang::exceptions::NoSuchElementException) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689).

If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

6.156.3.9 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::get (const K & *key*) throw (lang::exceptions::NoSuchElementException) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689).

If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1694).

6.156.3.10 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::isEmpty () const [inline, virtual]

Returns:

if the **Map** (p. 1689) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1695).

6.156.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::keySet () const [inline, virtual]

Returns a **Set** (p. 2320) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1490), **Set.remove** (p. 130), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns:

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1696).

6.156.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.156.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.156.3.14 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.156.3.15 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::put (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1697).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putAll(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::putIfAbsent(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::replace().

6.156.3.16 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

6.156.3.17 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putAll (const ConcurrentStlMap< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1689) in this one.

Parameters:

other A **Map** (p. 1689) instance whose elements are to all be inserted in this **Map** (p. 1689).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1697).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::copy().

6.156.3.18 template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::putIfAbsent (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

If the specified key is not already associated with a value, associate it with the given value. This is equivalent to

```
if( !map.containsKey( key ) ) {
    map.put( key, value );
    return true;
} else {
```

```
        return false;
    }
```

except that the action is performed atomically.

Parameters:

key The key to map the value to.

value The value to map to the given key.

Returns:

true if the put operation was performed otherwise return false which indicates there was a value previously mapped to the key.

Exceptions:

UnsupportedOperationException if the put operation is not supported by this map

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 914).

6.156.3.19 template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & *key*, const V & *value*) [inline, virtual]

Remove entry for key only if currently mapped to given value. Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == value ) ) {
    map.remove( key );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value associated with the specified key.

Returns:

true if the value was removed, false otherwise

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 915).

6.156.3.20 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::remove (const K & *key*) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 1689).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1698).

6.156.3.21 template<typename K, typename V, typename COMPARATOR = std::less<K>> V decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]

Replace entry for key only if currently mapped to some value. Acts as

```
if( map.containsKey( key ) ) {
    return map.put( key, value );
} else {
    throw NoSuchElementException(...);
};
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

value value to be associated with the specified key.

Returns:

copy of the previous value associated with specified key, or throws an NoSuchElementException if there was no mapping for key.

Exceptions:

NoSuchElementException if there was no previous mapping.

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 915).

6.156.3.22 template<typename K, typename V, typename COMPARATOR = std::less<K>> bool decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::replace (const K & *key*, const V & *oldValue*, const V & *newValue*) [inline, virtual]

Replace entry for key only if currently mapped to given value. Acts as

```
if( map.containsKey( key ) && ( map.get( key ) == oldValue ) ) {
    map.put( key, newValue );
    return true;
} else {
    return false;
}
```

except that the action is performed atomically.

Parameters:

key key with which the specified value is associated.

oldValue value expected to be associated with the specified key.

newValue value to be associated with the specified key.

Returns:

true if the value was replaced

Implements **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >** (p. 916).

6.156.3.23 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::size () const [inline, virtual]

Returns:

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1699).

6.156.3.24 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.156.3.25 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<V> decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::values () const [inline, virtual]

Returns:

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1700).

Referenced by decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::values().

6.156.3.26 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.156.3.27 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ConcurrentStlMap.h**

6.157 decaf::util::concurrent::locks::Condition Class Reference

Condition (p. 932) factors out the **Mutex** (p. 1921) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1618) implementations.

`#include <src/main/decaf/util/concurrent/locks/Condition.h>`

Public Member Functions

- virtual ~**Condition** ()
- virtual void **await** ()=0 throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)

  Causes the current thread to wait until it is signalled or interrupted.

- virtual void **awaitUninterruptibly** ()=0 throw (decaf::lang::exceptions::IllegalMonitorStateException)

  Causes the current thread to wait until it is signalled.

- virtual long long **awaitNanos** (long long nanosTimeout)=0 throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)

  Causes the current thread to wait until it is signalled or interrupted, or the specified waiting time elapses.

- virtual bool **await** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException)

  Causes the current thread to wait until it is signalled or interrupted, or the specified waiting time elapses.

- virtual void **signal** ()=0

  Wakes up one waiting thread.

- virtual void **signalAll** ()=0

  Wakes up all waiting threads.

6.157.1 Detailed Description

Condition (p. 932) factors out the **Mutex** (p. 1921) monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary **Lock** (p. 1618) implementations. Where a **Lock** (p. 1618) replaces the use of synchronized statements, a **Condition** (p. 932) replaces the use of the Object monitor methods.

Conditions (also known as condition queues or condition variables) provide a means for one thread to suspend execution (to "wait") until notified by another thread that some state condition may now be true. Because access to this shared state information occurs in different threads, it must be protected, so a lock of some form is associated with the condition. The key property that waiting

for a condition provides is that it atomically releases the associated lock and suspends the current thread.

A **Condition** (p. 932) instance is intrinsically bound to a lock. To obtain a **Condition** (p. 932) instance for a particular **Lock** (p. 1618) instance use its newCondition() method.

As an example, suppose we have a bounded buffer which supports put and take methods. If a take is attempted on an empty buffer, then the thread will block until an item becomes available; if a put is attempted on a full buffer, then the thread will block until a space becomes available. We would like to keep waiting put threads and take threads in separate wait-sets so that we can use the optimization of only notifying a single thread at a time when items or spaces become available in the buffer. This can be achieved using two **Condition** (p. 932) instances.

class BoundedBuffer { Lock* lock = new ReentrantLock(); Condition* notFull = lock->newCondition(); Condition* notEmpty = lock->newCondition();

Object* items = new Object[100]; int putptr, takeptr, count;

public void put(Object* x) throw(InterruptedException) { lock->lock(); try { while(count == 100) notFull->**await()** (p. 934); items[putptr] = x; if (++putptr == 100) putptr = 0; ++count; notEmpty->**signal()** (p. 937); } catch(...) { lock->unlock(); } }

public Object take() throw(InterruptedException) { lock->lock(); try { while(count == 0) notEmpty->**await()** (p. 934); Object x = items[takeptr]; if (++takeptr == 100) takeptr = 0; --count; notFull->**signal()** (p. 937); return x; } catch(...) { lock->unlock(); } } }

(The ArrayBlockingQueue class provides this functionality, so there is no reason to implement this sample usage class.)

Implementation Considerations

When waiting upon a **Condition** (p. 932), a "spurious wakeup" is permitted to occur, in general, as a concession to the underlying platform semantics. This has little practical impact on most application programs as a **Condition** (p. 932) should always be waited upon in a loop, testing the state predicate that is being waited for. An implementation is free to remove the possibility of spurious wakeups but it is recommended that applications programmers always assume that they can occur and so always wait in a loop.

The three forms of condition waiting (interruptible, non-interruptible, and timed) may differ in their ease of implementation on some platforms and in their performance characteristics. In particular, it may be difficult to provide these features and maintain specific semantics such as ordering guarantees. Further, the ability to interrupt the actual suspension of the thread may not always be feasible to implement on all platforms.

Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of waiting, nor is it required to support interruption of the actual suspension of the thread.

An implementation is required to clearly document the semantics and guarantees provided by each of the waiting methods, and when an implementation does support interruption of thread suspension then it must obey the interruption semantics as defined in this interface.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

6.157.2 Constructor & Destructor Documentation

6.157.2.1 virtual decaf::util::concurrent::locks::Condition::~Condition () [inline, virtual]

6.157.3 Member Function Documentation

6.157.3.1 virtual bool decaf::util::concurrent::locks::Condition::await (long long *time*, const TimeUnit & *unit*) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Causes the current thread to wait until it is signalled or interrupted, or the specified waiting time elapses. This method is behaviorally equivalent to:

awaitNanos(unit.toNanos(time)) > 0

Parameters:

time - the maximum time to wait

unit - the time unit of the time argument

Returns:

false if the waiting time detectably elapsed before return from the method, else true

Exceptions:

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.157.3.2 virtual void decaf::util::concurrent::locks::Condition::await () throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Causes the current thread to wait until it is signalled or interrupted. The lock associated with this **Condition** (p. 932) is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of four things happens:

∗ Some other thread invokes the **signal()** (p. 937) method for this **Condition** (p. 932) and the current thread happens to be chosen as the thread to be awakened; or ∗ Some other thread invokes the **signalAll()** (p. 937) method for this **Condition** (p. 932); or ∗ Some other thread interrupts the current thread, and interruption of thread suspension is supported; or ∗ A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

∗ has its interrupted status set on entry to this method; or ∗ is interrupted while waiting and interruption of thread suspension is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 932) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as IllegalMonitorStateException) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal. In that case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Exceptions:

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.157.3.3 virtual long long decaf::util::concurrent::locks::Condition::awaitNanos (long long *nanosTimeout*) throw (decaf::lang::exceptions::InterruptedException, decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Causes the current thread to wait until it is signalled or interrupted, or the specified waiting time elapses. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of five things happens:

∗ Some other thread invokes the **signal()** (p. 937) method for this **Condition** (p. 932) and the current thread happens to be chosen as the thread to be awakened; or ∗ Some other thread invokes the **signalAll()** (p. 937) method for this **Condition** (p. 932); or ∗ Some other thread interrupts the current thread, and interruption of thread suspension is supported; or ∗ The specified waiting time elapses; or ∗ A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread:

∗ has its interrupted status set on entry to this method; or ∗ is interrupted while waiting and interruption of thread suspension is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared. It is not specified, in the first case, whether or not the test for interruption occurs before the lock is released.

The method returns an estimate of the number of nanoseconds remaining to wait given the supplied nanosTimeout value upon return, or a value less than or equal to zero if it timed out. This value can be used to determine whether and how long to re-wait in cases where the wait returns but an awaited condition still does not hold. Typical uses of this method take the following form:

synchronized boolean aMethod(long timeout, const TimeUnit& unit) { long nanosTimeout = unit.toNanos(timeout); while (!conditionBeingWaitedFor) { if (nanosTimeout > 0) nanosTimeout = theCondition->awaitNanos(nanosTimeout); else return false; } // ... }

Design note: This method requires a nanosecond argument so as to avoid truncation errors in reporting remaining times. Such precision loss would make it difficult for programmers to ensure that total waiting times are not systematically shorter than specified when re-waits occur.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 932) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as IllegalMonitorStateException) and the implementation must document that fact.

An implementation can favor responding to an interrupt over normal method return in response to a signal, or over indicating the elapse of the specified waiting time. In either case the implementation must ensure that the signal is redirected to another waiting thread, if there is one.

Parameters:

nanosTimeout - the maximum time to wait, in nanoseconds

Returns:

an estimate of the nanosTimeout value minus the time spent waiting upon return from this method. A positive value may be used as the argument to a subsequent call to this method to finish waiting out the desired time. A value less than or equal to zero indicates that no time remains.

Exceptions:

InterruptedException if the current thread is interrupted (and interruption of thread suspension is supported)

IllegalMonitorStateException if the caller is not the lock owner.

6.157.3.4 virtual void decaf::util::concurrent::locks::Condition::awaitUninterruptibly () throw (decaf::lang::exceptions::IllegalMonitorStateException) [pure virtual]

Causes the current thread to wait until it is signalled. The lock associated with this condition is atomically released and the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

∗ Some other thread invokes the **signal()** (p. 937) method for this **Condition** (p. 932) and the current thread happens to be chosen as the thread to be awakened; or ∗ Some other thread invokes the **signalAll()** (p. 937) method for this **Condition** (p. 932); or ∗ A "spurious wakeup" occurs.

In all cases, before this method can return the current thread must re-acquire the lock associated with this condition. When the thread returns it is guaranteed to hold this lock.

If the current thread's interrupted status is set when it enters this method, or it is interrupted while waiting, it will continue to wait until signalled. When it finally returns from this method its interrupted status will still be set.

Implementation Considerations

The current thread is assumed to hold the lock associated with this **Condition** (p. 932) when this method is called. It is up to the implementation to determine if this is the case and if not, how to respond. Typically, an exception will be thrown (such as IllegalMonitorStateException) and the implementation must document that fact.

Exceptions:

IllegalMonitorStateException if the caller is not the lock owner.

6.157.3.5 virtual void decaf::util::concurrent::locks::Condition::signal () [pure virtual]

Wakes up one waiting thread. If any threads are waiting on this condition then one is selected for waking up. That thread must then re-acquire the lock before returning from await.

6.157.3.6 virtual void decaf::util::concurrent::locks::Condition::signalAll () [pure virtual]

Wakes up all waiting threads. If any threads are waiting on this condition then they are all woken up. Each thread must re-acquire the lock before it can return from await.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Condition.h**

6.158 decaf::net::ConnectException Class Reference

`#include <src/main/decaf/net/ConnectException.h>`Inheritance diagram for decaf::net::ConnectException:

Public Member Functions

- **ConnectException** () throw ()

  Default Constructor.

- **ConnectException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **ConnectException** (const **ConnectException** &ex) throw ()

  Copy Constructor.

- **ConnectException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **ConnectException** (const std::exception ***cause**) throw ()

  Constructor.

- **ConnectException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual **ConnectException** * **clone** () const

  Clones this exception.

- virtual ~**ConnectException** () throw ()

6.158.1 Constructor & Destructor Documentation

6.158.1.1 decaf::net::ConnectException::ConnectException () throw () `[inline]`

Default Constructor.

6.158.1.2 decaf::net::ConnectException::ConnectException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.158.1.3 decaf::net::ConnectException::ConnectException (const ConnectException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.158.1.4 decaf::net::ConnectException::ConnectException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.158.1.5 decaf::net::ConnectException::ConnectException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.158.1.6 decaf::net::ConnectException::ConnectException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.158.1.7 virtual decaf::net::ConnectException::~ConnectException () throw () [inline, virtual]

6.158.2 Member Function Documentation

6.158.2.1 virtual ConnectException* decaf::net::ConnectException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2380).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ConnectException.h**

6.159 cms::Connection Class Reference

The client's connection to its provider.

`#include <src/main/cms/Connection.h>`Inheritance diagram for cms::Connection:

Public Member Functions

- virtual ~**Connection** ()
- virtual void **close** ()=0 throw (CMSException)

  Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

- virtual const **ConnectionMetaData** * **getMetaData** () const =0 throw (CMSException)

  Gets the metadata for this connection.

- virtual **Session** * **createSession** ()=0 throw (CMSException)

  *Creates an AUTO_ACKNOWLEDGE **Session** (p. 2270).*

- virtual **Session** * **createSession** (**Session::AcknowledgeMode** ackMode)=0 throw (CMSException)

  *Creates a new **Session** (p. 2270) to work for this **Connection** (p. 941) using the specified acknowledgment mode.*

- virtual std::string **getClientID** () const =0

  Get the Client Id for this session.

- virtual **ExceptionListener** * **getExceptionListener** () const =0

  Gets the registered Exception Listener for this connection.

- virtual void **setExceptionListener** (**ExceptionListener** *listener)=0

  Sets the registered Exception Listener for this connection.

6.159.1 Detailed Description

The client's connection to its provider. Connections support concurrent use.

A connection serves several purposes:

- It encapsulates an open connection with a JMS provider. It typically represents an open TCP/IP socket between a client and the service provider software.
- Its creation is where client authentication takes place.
- It can specify a unique client identifier.
- It provides a **ConnectionMetaData** (p. 1015) object.
- It supports an optional **ExceptionListener** (p. 1275) object.

Because the creation of a connection involves setting up authentication and communication, a connection is a relatively heavy-weight object. Most clients will do all their messaging with a single connection. Other more advanced applications may use several connections. The CMS API does not architect a reason for using multiple connections; however, there may be operational reasons for doing so.

A CMS client typically creates a connection, one or more sessions, and a number of message producers and consumers. When a connection is created, it is in stopped mode. That means that no messages are being delivered.

It is typical to leave the connection in stopped mode until setup is complete (that is, until all message consumers have been created). At that point, the client calls the connection's start method, and messages begin arriving at the connection's consumers. This setup convention minimizes any client confusion that may result from asynchronous message delivery while the client is still in the process of setting itself up.

A connection can be started immediately, and the setup can be done afterwards. Clients that do this must be prepared to handle asynchronous message delivery while they are still in the process of setting up.

A message producer can send messages while a connection is stopped.

Since:

1.0

6.159.2 Constructor & Destructor Documentation

6.159.2.1 virtual cms::Connection::~Connection () [inline, virtual]

6.159.3 Member Function Documentation

6.159.3.1 virtual void cms::Connection::close () throw (CMSException) [pure virtual]

Closes this connection as well as any Sessions created from it (and those Sessions' consumers and producers).

Exceptions:

CMSException *(*p. ***850)***

Implements **cms::Closeable** (p. 838).

Implemented in **activemq::core::ActiveMQConnection** (p. 193).

6.159.3.2 virtual Session* cms::Connection::createSession (Session::AcknowledgeMode *ackMode*) throw (CMSException) [pure virtual]

Creates a new **Session** (p. 2270) to work for this **Connection** (p. 941) using the specified acknowledgment mode.

Parameters:

ackMode the Acknowledgment Mode to use.

Exceptions:

CMSException (p. ***850***)

Implemented in **activemq::core::ActiveMQConnection** (p. 193).

6.159.3.3 virtual Session* cms::Connection::createSession () throw (CMSException) [pure virtual]

Creates an AUTO_ACKNOWLEDGE **Session** (p. 2270).

Exceptions:

CMSException (p. ***850***)

Implemented in **activemq::core::ActiveMQConnection** (p. 194).

6.159.3.4 virtual std::string cms::Connection::getClientID () const [pure virtual]

Get the Client Id for this session.

Returns:

Client Id String

Implemented in **activemq::core::ActiveMQConnection** (p. 195).

6.159.3.5 virtual ExceptionListener* cms::Connection::getExceptionListener () const [pure virtual]

Gets the registered Exception Listener for this connection.

Returns:

pointer to an exception listener or NULL

Implemented in **activemq::core::ActiveMQConnection** (p. 196).

6.159.3.6 virtual const ConnectionMetaData* cms::Connection::getMetaData () const throw (CMSException) [pure virtual]

Gets the metadata for this connection.

Returns:

the connection MetaData pointer (caller does not own it).

Exceptions:

CMSException (p. ***850***) if the provider fails to get the connection metadata for this connection.

See also:

ConnectionMetaData (p. 1015)

Since:

2.0

Implemented in **activemq::core::ActiveMQConnection** (p. 196).

6.159.3.7 virtual void cms::Connection::setExceptionListener (ExceptionListener * *listener*) [pure virtual]

Sets the registered Exception Listener for this connection.

Parameters:

listener pointer to and `ExceptionListener` (p. 1275)

Implemented in **activemq::core::ActiveMQConnection** (p. 198).

The documentation for this class was generated from the following file:

- src/main/cms/**Connection.h**

6.160 activemq::commands::ConnectionControl Class Reference

`#include <src/main/activemq/commands/ConnectionControl.h>`Inheritance diagram for activemq::commands::ConnectionControl:

Public Member Functions

- **ConnectionControl** ()
- virtual ~**ConnectionControl** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionControl** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual bool **isClose** () const
- virtual void **setClose** (bool **close**)
- virtual bool **isExit** () const
- virtual void **setExit** (bool **exit**)
- virtual bool **isFaultTolerant** () const
- virtual void **setFaultTolerant** (bool **faultTolerant**)
- virtual bool **isResume** () const
- virtual void **setResume** (bool **resume**)
- virtual bool **isSuspend** () const
- virtual void **setSuspend** (bool **suspend**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONCONTROL** = 18

Protected Member Functions

- **ConnectionControl** (const **ConnectionControl** &)
- **ConnectionControl** & **operator=** (const **ConnectionControl** &)

Protected Attributes

- bool **close**
- bool **exit**
- bool **faultTolerant**
- bool **resume**
- bool **suspend**

6.160.1 Constructor & Destructor Documentation

6.160.1.1 activemq::commands::ConnectionControl::ConnectionControl (const ConnectionControl &) [inline, protected]

6.160.1.2 activemq::commands::ConnectionControl::ConnectionControl ()

6.160.1.3 virtual activemq::commands::ConnectionControl::~ConnectionControl () [virtual]

6.160.2 Member Function Documentation

6.160.2.1 virtual ConnectionControl* activemq::commands::ConnectionControl::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.160.2.2 virtual void activemq::commands::ConnectionControl::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.160.2.3 virtual bool activemq::commands::ConnectionControl::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.160.2.4 virtual unsigned char activemq::commands::ConnectionControl::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.160.2.5 virtual bool activemq::commands::ConnectionControl::isClose () const [virtual]

6.160.2.6 virtual bool activemq::commands::ConnectionControl::isExit () const [virtual]

6.160.2.7 virtual bool activemq::commands::ConnectionControl::isFaultTolerant () const [virtual]

6.160.2.8 virtual bool activemq::commands::ConnectionControl::isResume () const [virtual]

6.160.2.9 virtual bool activemq::commands::ConnectionControl::isSuspend () const [virtual]

6.160.2.10 ConnectionControl& activemq::commands::ConnectionControl::operator= (const ConnectionControl &) [inline, protected]

6.160.2.11 virtual void activemq::commands::ConnectionControl::setClose (bool *close*) [virtual]

6.160.2.12 virtual void activemq::commands::ConnectionControl::setExit (bool *exit*) [virtual]

6.160.2.13 virtual void activemq::commands::ConnectionControl::setFaultTolerant (bool *faultTolerant*) [virtual]

6.160.2.14 virtual void activemq::commands::ConnectionControl::setResume (bool *resume*) [virtual]

6.160.2.15 virtual void activemq::commands::ConnectionControl::setSuspend (bool *suspend*) [virtual]

6.160.2.16 virtual std::string activemq::commands::ConnectionControl::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.160.2.17 virtual Pointer<Command> activemq::commands::ConnectionControl::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.160.3 Field Documentation

6.160.3.1 bool activemq::commands::ConnectionControl::close [protected]

6.160.3.2 bool activemq::commands::ConnectionControl::exit [protected]

6.160.3.3 bool activemq::commands::ConnectionControl::faultTolerant [protected]

6.160.3.4 const unsigned char activemq::commands::ConnectionControl::ID_CONNECTIONCONTROL = 18 [static]

6.160.3.5 bool activemq::commands::ConnectionControl::resume [protected]

6.160.3.6 bool activemq::commands::ConnectionControl::suspend [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionControl.h**

6.161 activemq::wireformat::openwire::marshal::v3::ConnectionControlM Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 950).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual ~**ConnectionControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.161.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 950). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.161.2 Constructor & Destructor Documentation

6.161.2.1 activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::ConnectionC () [inline]

6.161.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::~Connection () [inline, virtual]

6.161.3 Member Function Documentation

6.161.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.161.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.161.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.161.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.161.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.161.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.161.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionControlMarshaller.h**

6.162 activemq::wireformat::openwire::marshal::v1::ConnectionControlM Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 954).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual ~**ConnectionControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.162.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 954). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.162.2 Constructor & Destructor Documentation

6.162.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::ConnectionC () [inline]

6.162.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::~Connection () [inline, virtual]

6.162.3 Member Function Documentation

6.162.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.162.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.162.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.162.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.162.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.162.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.162.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionControlMarshaller.h**

6.163 activemq::wireformat::openwire::marshal::v2::ConnectionControlM Class Reference

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 958).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionControlMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller:

Public Member Functions

- **ConnectionControlMarshaller** ()
- virtual ~**ConnectionControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.163.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionControlMarshaller** (p. 958). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.163.2 Constructor & Destructor Documentation

6.163.2.1 activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::ConnectionC () [inline]

6.163.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::~Connection () [inline, virtual]

6.163.3 Member Function Documentation

6.163.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.163.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.163.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.163.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.163.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.163.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.163.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionControlMarshaller.h**

6.164 activemq::commands::ConnectionError Class Reference

`#include <src/main/activemq/commands/ConnectionError.h>`Inheritance diagram for activemq::commands::ConnectionError:

Public Member Functions

- **ConnectionError** ()
- virtual ~**ConnectionError** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionError** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &**exception**)
- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &**connectionId**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONERROR** = 16

Protected Member Functions

- **ConnectionError** (const **ConnectionError** &)
- **ConnectionError** & **operator**= (const **ConnectionError** &)

Protected Attributes

- **Pointer< BrokerError > exception**
- **Pointer< ConnectionId > connectionId**

6.164.1 Constructor & Destructor Documentation

6.164.1.1 activemq::commands::ConnectionError::ConnectionError (const ConnectionError &) [inline, protected]

6.164.1.2 activemq::commands::ConnectionError::ConnectionError ()

6.164.1.3 virtual activemq::commands::ConnectionError::~ConnectionError () [virtual]

6.164.2 Member Function Documentation

6.164.2.1 virtual ConnectionError* activemq::commands::ConnectionError::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.164.2.2 virtual void activemq::commands::ConnectionError::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.164.2.3 virtual bool activemq::commands::ConnectionError::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.164.2.4 virtual Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () [virtual]

6.164.2.5 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionError::getConnectionId () const [virtual]

6.164.2.6 virtual unsigned char activemq::commands::ConnectionError::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.164.2.7 virtual Pointer<BrokerError>& activemq::commands::ConnectionError::getException () [virtual]

6.164.2.8 virtual const Pointer<BrokerError>& activemq::commands::ConnectionError::getException () const [virtual]

6.164.2.9 ConnectionError& activemq::commands::ConnectionError::operator= (const ConnectionError &) [inline, protected]

6.164.2.10 virtual void activemq::commands::ConnectionError::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.164.2.11 virtual void activemq::commands::ConnectionError::setException (const Pointer< BrokerError > & *exception*) [virtual]

6.164.2.12 virtual std::string activemq::commands::ConnectionError::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.164.2.13 virtual Pointer<Command> activemq::commands::ConnectionError::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.164.3 Field Documentation

6.164.3.1 Pointer<ConnectionId> activemq::commands::ConnectionError::connectionId [protected]

6.164.3.2 Pointer<BrokerError> activemq::commands::ConnectionError::exception [protected]

6.164.3.3 const unsigned char activemq::commands::ConnectionError::ID_CONNECTIONERROR = 16 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionError.h**

6.165 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 966).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionErrorMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual ~**ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.165.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 966). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.165.2 Constructor & Destructor Documentation

6.165.2.1 activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::ConnectionErro () [inline]

6.165.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::~ConnectionEr () [inline, virtual]

6.165.3 Member Function Documentation

6.165.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.165.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.165.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.165.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.165.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.165.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.165.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionErrorMarshaller.h**

6.166 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 970).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionErrorMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual ~**ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.166.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 970). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.166.2 Constructor & Destructor Documentation

6.166.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::ConnectionErro () [inline]

6.166.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::~ConnectionEr () [inline, virtual]

6.166.3 Member Function Documentation

6.166.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.166.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.166.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.166.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.166.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.166.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.166.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionErrorMarshaller.h**

6.167 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMa Class Reference

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 974).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionErrorMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller:

Public Member Functions

- **ConnectionErrorMarshaller** ()
- virtual ~**ConnectionErrorMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.167.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionErrorMarshaller** (p. 974). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.167.2 Constructor & Destructor Documentation

6.167.2.1 activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::ConnectionErro () [inline]

6.167.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::~ConnectionEr () [inline, virtual]

6.167.3 Member Function Documentation

6.167.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.167.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.167.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.167.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.167.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.167.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.167.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionErrorMarshaller.h**

6.168 cms::ConnectionFactory Class Reference

Defines the interface for a factory that creates connection objects, the **Connection** (p. 941) objects returned implement the CMS **Connection** (p. 941) interface and hide the CMS Provider specific implementation details behind that interface.

`#include <src/main/cms/ConnectionFactory.h>`Inheritance diagram for cms::ConnectionFactory:

Public Member Functions

- virtual ~**ConnectionFactory** ()
- virtual **Connection** * **createConnection** ()=0 throw (CMSException)

  Creates a connection with the default user identity.

- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password)=0 throw (cms::CMSException)

  Creates a connection with the default specified identity.

- virtual **cms::Connection** * **createConnection** (const std::string &username, const std::string &password, const std::string &clientId)=0 throw (cms::CMSException)

  Creates a connection with the specified user identity.

Static Public Member Functions

- static **ConnectionFactory** * **createCMSConnectionFactory** (const std::string &brokerURI) throw (cms::CMSException)

  Static method that is used to create a provider specific connection factory.

6.168.1 Detailed Description

Defines the interface for a factory that creates connection objects, the **Connection** (p. 941) objects returned implement the CMS **Connection** (p. 941) interface and hide the CMS Provider specific implementation details behind that interface. A Client creates a new **ConnectionFactory** (p. 978) either directly by instantiating the provider specific implementation of the factory or by using the static method `createCMSConnectionFactory` which all providers are required to implement.

Since:

1.0

6.168.2 Constructor & Destructor Documentation

6.168.2.1 virtual cms::ConnectionFactory::~ConnectionFactory () [inline, virtual]

6.168.3 Member Function Documentation

6.168.3.1 static ConnectionFactory* cms::ConnectionFactory::createCMSConnectionFactory (const std::string & *brokerURI*) throw (cms::CMSException) [static]

Static method that is used to create a provider specific connection factory. The provider implements this method in their library and returns an instance of a **ConnectionFactory** (p. 978) derived object. Clients can use this method to remain abstracted from the specific CMS implementation being used.

Parameters:

brokerURI The remote address to use to connect to the Provider.

Returns:

A pointer to a provider specific implementation of the **ConnectionFactory** (p. 978) interface, the caller is responsible for deleting this resource.

Exceptions:

CMSException *(p. 850)* if an internal error occurs while creating the **ConnectionFactory** (p. 978).

6.168.3.2 virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*, const std::string & *clientId*) throw (cms::CMSException) [pure virtual]

Creates a connection with the specified user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2413) method is explicitly called. The user name and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

clientId The Client Id assigned to connection. If the id is the empty string ("") then a random client Id is created for this connection.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSException *(p. 850)* if an internal error occurs while creating the **Connection** (p. 941).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 202).

6.168.3.3 virtual cms::Connection* cms::ConnectionFactory::createConnection (const std::string & *username*, const std::string & *password*) throw (cms::CMSException) [pure virtual]

Creates a connection with the default specified identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2413) method is explicitly called. The username and password values passed here do not change the defaults, subsequent calls to the parameterless createConnection will continue to use the default values that were set in the Constructor.

Parameters:

username The user name used to authenticate with the Provider.

password The password used to authenticate with the Provider.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSException (p. ***850***) if an internal error occurs while creating the **Connection** (p. 941).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 202).

6.168.3.4 virtual Connection* cms::ConnectionFactory::createConnection () throw (CMSException) [pure virtual]

Creates a connection with the default user identity. The connection is created in stopped mode. No messages will be delivered until the **Connection.start** (p. 2413) method is explicitly called.

Returns:

A pointer to a connection object, caller owns the pointer and is responsible for closing the connection and deleting the instance.

Exceptions:

CMSException (p. ***850***) if an internal error occurs while creating the **Connection** (p. 941).

Implemented in **activemq::core::ActiveMQConnectionFactory** (p. 202).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionFactory.h**

6.169 activemq::commands::ConnectionId Class Reference

`#include <src/main/activemq/commands/ConnectionId.h>`Inheritance diagram for activemq::commands::ConnectionId:

Public Types

- typedef **decaf::lang::PointerComparator< ConnectionId > COMPARATOR**

Public Member Functions

- **ConnectionId** ()
- **ConnectionId** (const **ConnectionId** &other)
- **ConnectionId** (const **SessionId** *sessionId)
- **ConnectionId** (const **ProducerId** *producerId)
- **ConnectionId** (const **ConsumerId** *consumerId)
- virtual ~**ConnectionId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getValue** () const
- virtual std::string & **getValue** ()
- virtual void **setValue** (const std::string &**value**)
- virtual int **compareTo** (const **ConnectionId** &**value**) const
- virtual bool **equals** (const **ConnectionId** &**value**) const
- virtual bool **operator==** (const **ConnectionId** &**value**) const
- virtual bool **operator<** (const **ConnectionId** &**value**) const
- **ConnectionId** & **operator=** (const **ConnectionId** &other)

Static Public Attributes

- static const unsigned char **ID_CONNECTIONID** = 120

Protected Attributes

- std::string **value**

6.169.1 Member Typedef Documentation

6.169.1.1 typedef decaf::lang::PointerComparator<ConnectionId> activemq::commands::ConnectionId::COMPARATOR

6.169.2 Constructor & Destructor Documentation

6.169.2.1 activemq::commands::ConnectionId::ConnectionId ()

6.169.2.2 activemq::commands::ConnectionId::ConnectionId (const ConnectionId & *other*)

6.169.2.3 activemq::commands::ConnectionId::ConnectionId (const SessionId * *sessionId*)

6.169.2.4 activemq::commands::ConnectionId::ConnectionId (const ProducerId * *producerId*)

6.169.2.5 activemq::commands::ConnectionId::ConnectionId (const ConsumerId * *consumerId*)

6.169.2.6 virtual activemq::commands::ConnectionId::~ConnectionId () [virtual]

6.169.3 Member Function Documentation

6.169.3.1 virtual ConnectionId* activemq::commands::ConnectionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.169.3.2 virtual int activemq::commands::ConnectionId::compareTo (const ConnectionId & *value*) const [virtual]

6.169.3.3 virtual void activemq::commands::ConnectionId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.169.3.4 virtual bool activemq::commands::ConnectionId::equals (const ConnectionId & *value*) const [virtual]

6.169.3.5 virtual bool activemq::commands::ConnectionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.169.3.6 virtual unsigned char activemq::commands::ConnectionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.169.3.7 virtual std::string& activemq::commands::ConnectionId::getValue () [virtual]

6.169.3.8 virtual const std::string& activemq::commands::ConnectionId::getValue () const [virtual]

6.169.3.9 virtual bool activemq::commands::ConnectionId::operator< (const ConnectionId & *value*) const [virtual]

6.169.3.10 ConnectionId& activemq::commands::ConnectionId::operator= (const ConnectionId & *other*)

6.169.3.11 virtual bool activemq::commands::ConnectionId::operator== (const ConnectionId & *value*) const [virtual]

6.169.3.12 virtual void activemq::commands::ConnectionId::setValue (const std::string & *value*) [virtual]

6.169.3.13 virtual std::string activemq::commands::ConnectionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.169.4 Field Documentation

6.169.4.1 const unsigned char activemq::commands::ConnectionId::ID_CONNECTIONID = 120 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.169.4.2 std::string activemq::commands::ConnectionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionId.h**

6.170 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarsha Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 985).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual ~**ConnectionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.170.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 985). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.170.2 Constructor & Destructor Documentation

6.170.2.1 activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::ConnectionIdMarsh () [inline]

6.170.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::~ConnectionIdMar () [inline, virtual]

6.170.3 Member Function Documentation

6.170.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.170.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.170.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.170.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.170.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.170.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.170.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionIdMarshaller.h**

6.171 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarsha Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 989).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual ~**ConnectionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.171.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 989). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.171.2 Constructor & Destructor Documentation

6.171.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::ConnectionIdMarsh () [inline]

6.171.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::~ConnectionIdMar () [inline, virtual]

6.171.3 Member Function Documentation

6.171.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.171.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.171.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.171.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.171.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.171.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.171.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionIdMarshaller.h**

6.172 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarsha Class Reference

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 993).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller:

Public Member Functions

- **ConnectionIdMarshaller** ()
- virtual ~**ConnectionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.172.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionIdMarshaller** (p. 993). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.172.2 Constructor & Destructor Documentation

6.172.2.1 activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::ConnectionIdMarsh () [inline]

6.172.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::~ConnectionIdMar () [inline, virtual]

6.172.3 Member Function Documentation

6.172.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.172.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.172.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.172.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.172.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.172.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.172.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionIdMarshaller.h**

6.173 activemq::commands::ConnectionInfo Class Reference

`#include <src/main/activemq/commands/ConnectionInfo.h>`Inheritance diagram for activemq::commands::ConnectionInfo:

Public Member Functions

- **ConnectionInfo** ()
- virtual ~**ConnectionInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConnectionInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &**connectionId**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &**clientId**)
- virtual const std::string & **getPassword** () const
- virtual std::string & **getPassword** ()
- virtual void **setPassword** (const std::string &**password**)
- virtual const std::string & **getUserName** () const
- virtual std::string & **getUserName** ()
- virtual void **setUserName** (const std::string &**userName**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual bool **isBrokerMasterConnector** () const
- virtual void **setBrokerMasterConnector** (bool **brokerMasterConnector**)
- virtual bool **isManageable** () const
- virtual void **setManageable** (bool **manageable**)
- virtual bool **isClientMaster** () const

- virtual void **setClientMaster** (bool **clientMaster**)
- virtual bool **isConnectionInfo** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONNECTIONINFO** = 3

Protected Member Functions

- **ConnectionInfo** (const **ConnectionInfo** &)
- **ConnectionInfo** & **operator=** (const **ConnectionInfo** &)

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **clientId**
- std::string **password**
- std::string **userName**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- bool **brokerMasterConnector**
- bool **manageable**
- bool **clientMaster**

6.173.1 Constructor & Destructor Documentation

6.173.1.1 activemq::commands::ConnectionInfo::ConnectionInfo (const ConnectionInfo &) [inline, protected]

6.173.1.2 activemq::commands::ConnectionInfo::ConnectionInfo ()

6.173.1.3 virtual activemq::commands::ConnectionInfo::~ConnectionInfo () [virtual]

6.173.2 Member Function Documentation

6.173.2.1 virtual ConnectionInfo* activemq::commands::ConnectionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.173.2.2 virtual void activemq::commands::ConnectionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.173.2.3 virtual bool activemq::commands::ConnectionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.173.2.4 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () [virtual]

6.173.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConnectionInfo::getBrokerPath () const [virtual]

6.173.2.6 virtual std::string& activemq::commands::ConnectionInfo::getClientId () [virtual]

6.173.2.7 virtual const std::string& activemq::commands::ConnectionInfo::getClientId () const [virtual]

6.173.2.8 virtual Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () [virtual]

6.173.2.9 virtual const Pointer<ConnectionId>& activemq::commands::ConnectionInfo::getConnectionId () const [virtual]

6.173.2.10 virtual unsigned char activemq::commands::ConnectionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.173.2.11 virtual std::string& activemq::commands::ConnectionInfo::getPassword () [virtual]

6.173.2.12 virtual const std::string& activemq::commands::ConnectionInfo::getPassword () const [virtual]

6.173.2.13 virtual std::string& activemq::commands::ConnectionInfo::getUserName () [virtual]

6.173.2.14 virtual const std::string& activemq::commands::ConnectionInfo::getUserName () const [virtual]

6.173.2.15 virtual bool activemq::commands::ConnectionInfo::isBrokerMasterConnector () const [virtual]

6.173.2.16 virtual bool activemq::commands::ConnectionInfo::isClientMaster () const [virtual]

6.173.2.17 virtual bool activemq::commands::ConnectionInfo::isConnectionInfo () const [inline, virtual]

Returns:

an answer of true to the **isConnectionInfo()** (p. 1000) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.173.2.18 virtual bool activemq::commands::ConnectionInfo::isManageable () const [virtual]

6.173.2.19 ConnectionInfo& activemq::commands::ConnectionInfo::operator= (const ConnectionInfo &) [inline, protected]

6.173.2.20 virtual void activemq::commands::ConnectionInfo::setBrokerMasterConnector (bool *brokerMasterConnector*) [virtual]

6.173.2.21 virtual void activemq::commands::ConnectionInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.173.2.22 virtual void activemq::commands::ConnectionInfo::setClientId (const std::string & *clientId*) [virtual]

6.173.2.23 virtual void activemq::commands::ConnectionInfo::setClientMaster (bool *clientMaster*) [virtual]

6.173.2.24 virtual void activemq::commands::ConnectionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.173.2.25 virtual void activemq::commands::ConnectionInfo::setManageable (bool *manageable*) [virtual]

6.173.2.26 virtual void activemq::commands::ConnectionInfo::setPassword (const std::string & *password*) [virtual]

6.173.2.27 virtual void activemq::commands::ConnectionInfo::setUserName (const std::string & *userName*) [virtual]

6.173.2.28 virtual std::string activemq::commands::ConnectionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.173.2.29 virtual Pointer<Command> activemq::commands::ConnectionInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.173.3 Field Documentation

6.173.3.1 bool activemq::commands::ConnectionInfo::brokerMasterConnector [protected]

6.173.3.2 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConnectionInfo::brokerPath [protected]

6.173.3.3 std::string activemq::commands::ConnectionInfo::clientId [protected]

6.173.3.4 bool activemq::commands::ConnectionInfo::clientMaster [protected]

6.173.3.5 Pointer<ConnectionId> activemq::commands::ConnectionInfo::connectionId [protected]

6.173.3.6 const unsigned char activemq::commands::ConnectionInfo::ID_CONNECTIONINFO = 3 [static]

6.173.3.7 bool activemq::commands::ConnectionInfo::manageable [protected]

6.173.3.8 std::string activemq::commands::ConnectionInfo::password [protected]

6.173.3.9 std::string activemq::commands::ConnectionInfo::userName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConnectionInfo.h**

6.174 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMars Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1003).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConnectionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual ~**ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.174.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1003). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.174.2 Constructor & Destructor Documentation

6.174.2.1 activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::ConnectionInfoM () [inline]

6.174.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::~ConnectionInfo () [inline, virtual]

6.174.3 Member Function Documentation

6.174.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.174.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.174.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.174.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.174.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.174.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.174.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConnectionInfoMarshaller.h**

6.175 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMars Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1007).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConnectionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual ~**ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.175.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1007). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.175.2 Constructor & Destructor Documentation

6.175.2.1 activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::ConnectionInfoM () [inline]

6.175.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::~ConnectionInfo () [inline, virtual]

6.175.3 Member Function Documentation

6.175.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.175.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.175.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.175.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.175.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.175.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.175.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConnectionInfoMarshaller.h**

6.176 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMars Class Reference

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1011).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConnectionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller:

Public Member Functions

- **ConnectionInfoMarshaller** ()
- virtual ~**ConnectionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.176.1 Detailed Description

Marshaling code for Open Wire Format for **ConnectionInfoMarshaller** (p. 1011). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.176.2 Constructor & Destructor Documentation

6.176.2.1 activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::ConnectionInfoM () [inline]

6.176.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::~ConnectionInfo () [inline, virtual]

6.176.3 Member Function Documentation

6.176.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.176.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.176.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.176.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.176.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.176.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.176.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConnectionInfoMarshaller.h**

6.177 cms::ConnectionMetaData Class Reference

A **ConnectionMetaData** (p. 1015) object provides information describing the **Connection** (p. 941) object.

`#include <src/main/cms/ConnectionMetaData.h>`Inheritance diagram for cms::ConnectionMetaData:

Public Member Functions

- virtual ~**ConnectionMetaData** ()
- virtual std::string **getCMSVersion** () const =0 throw (cms::CMSException)

  Gets the CMS API version.

- virtual int **getCMSMajorVersion** () const =0 throw (cms::CMSException)

  Gets the CMS major version number.

- virtual int **getCMSMinorVersion** () const =0 throw (cms::CMSException)

  Gets the CMS minor version number.

- virtual std::string **getCMSProviderName** () const =0 throw (cms::CMSException)

  Gets the CMS provider name.

- virtual std::string **getProviderVersion** () const =0 throw (cms::CMSException)

  Gets the CMS provider version.

- virtual int **getProviderMajorVersion** () const =0 throw (cms::CMSException)

  Gets the CMS provider major version number.

- virtual int **getProviderMinorVersion** () const =0 throw (cms::CMSException)

  Gets the CMS provider minor version number.

- virtual std::vector< std::string > **getCMSXPropertyNames** () const =0 throw (cms::CMSException)

  Gets an Vector of the CMSX property names.

6.177.1 Detailed Description

A **ConnectionMetaData** (p. 1015) object provides information describing the **Connection** (p. 941) object.

Since:

1.3

6.177.2 Constructor & Destructor Documentation

6.177.2.1 virtual cms::ConnectionMetaData::~ConnectionMetaData () [inline, virtual]

6.177.3 Member Function Documentation

6.177.3.1 virtual int cms::ConnectionMetaData::getCMSMajorVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS major version number.

Returns:

the CMS API major version number

Exceptions:

CMSException *(*p. *850)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 206).

6.177.3.2 virtual int cms::ConnectionMetaData::getCMSMinorVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS minor version number.

Returns:

the CMS API minor version number

Exceptions:

CMSException *(*p. *850)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 206).

6.177.3.3 virtual std::string cms::ConnectionMetaData::getCMSProviderName () const throw (cms::CMSException) [pure virtual]

Gets the CMS provider name.

Returns:

the CMS provider name

Exceptions:

CMSException *(*p. *850)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 206).

6.177.3.4 virtual std::string cms::ConnectionMetaData::getCMSVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS API version.

Returns:

the CMS API Version in String form.

Exceptions:

CMSException *(*p. ***850****)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 207).

6.177.3.5 virtual std::vector<std::string> cms::ConnectionMetaData::getCMSXPropertyNames () const throw (cms::CMSException) [pure virtual]

Gets an Vector of the CMSX property names.

Returns:

an Vector of CMSX property names

Exceptions:

CMSException *(*p. ***850****)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 207).

6.177.3.6 virtual int cms::ConnectionMetaData::getProviderMajorVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS provider major version number.

Returns:

the CMS provider major version number

Exceptions:

CMSException *(*p. ***850****)* If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 207).

6.177.3.7 virtual int cms::ConnectionMetaData::getProviderMinorVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS provider minor version number.

Returns:

the CMS provider minor version number

Exceptions:

CMSException *(*p. ***850)*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 208).

6.177.3.8 virtual std::string cms::ConnectionMetaData::getProviderVersion () const throw (cms::CMSException) [pure virtual]

Gets the CMS provider version.

Returns:

the CMS provider version

Exceptions:

CMSException *(*p. ***850)*** If the CMS Provider fails to retrieve the metadata due to some internal error.

Implemented in **activemq::core::ActiveMQConnectionMetaData** (p. 208).

The documentation for this class was generated from the following file:

- src/main/cms/**ConnectionMetaData.h**

6.178 activemq::state::ConnectionState Class Reference

`#include <src/main/activemq/state/ConnectionState.h>`

Public Member Functions

- **ConnectionState** (const **Pointer**< **ConnectionInfo** > &info)
- virtual ~**ConnectionState** ()
- std::string **toString** () const
- const **Pointer**< **commands::ConnectionInfo** > & **getInfo** () const
- void **checkShutdown** () const
- void **shutdown** ()
- void **reset** (const **Pointer**< **ConnectionInfo** > &info)
- void **addTempDestination** (const **Pointer**< **DestinationInfo** > &info)
- void **removeTempDestination** (const **Pointer**< **ActiveMQDestination** > &destination)
- void **addTransactionState** (const **Pointer**< **TransactionId** > &id)
- const **Pointer**< **TransactionState** > & **getTransactionState** (const **Pointer**< **TransactionId** > &id) const
- std::vector< **Pointer**< **TransactionState** > > **getTransactionStates** () const
- **Pointer**< **TransactionState** > **removeTransactionState** (const **Pointer**< **TransactionId** > &id)
- void **addSession** (const **Pointer**< **SessionInfo** > &info)
- **Pointer**< **SessionState** > **removeSession** (const **Pointer**< **SessionId** > &id)
- const **Pointer**< **SessionState** > & **getSessionState** (const **Pointer**< **SessionId** > &id) const
- const **StlList**< **Pointer**< **DestinationInfo** > > & **getTempDesinations** () const
- std::vector< **Pointer**< **SessionState** > > **getSessionStates** () const

6.178.1 Constructor & Destructor Documentation

6.178.1.1 activemq::state::ConnectionState::ConnectionState (const Pointer< ConnectionInfo > & *info*)

6.178.1.2 virtual activemq::state::ConnectionState::~ConnectionState () [virtual]

6.178.2 Member Function Documentation

6.178.2.1 void activemq::state::ConnectionState::addSession (const Pointer< SessionInfo > & *info*) [inline]

6.178.2.2 void activemq::state::ConnectionState::addTempDestination (const Pointer< DestinationInfo > & *info*) [inline]

6.178.2.3 void activemq::state::ConnectionState::addTransactionState (const Pointer< TransactionId > & *id*) [inline]

6.178.2.4 void activemq::state::ConnectionState::checkShutdown () const

6.178.2.5 const Pointer<commands::ConnectionInfo>& activemq::state::ConnectionState::getInfo () const [inline]

6.178.2.6 const Pointer<SessionState>& activemq::state::ConnectionState::getSessionState (const Pointer< SessionId > & *id*) const [inline]

6.178.2.7 std::vector< Pointer<SessionState> > activemq::state::ConnectionState::getSessionStates () const [inline]

6.178.2.8 const StlList< Pointer<DestinationInfo> >& activemq::state::ConnectionState::getTempDesinations () const [inline]

6.178.2.9 const Pointer<TransactionState>& activemq::state::ConnectionState::getTransactionState (const Pointer< TransactionId > & *id*) const [inline]

6.178.2.10 std::vector< Pointer<TransactionState> > activemq::state::ConnectionState::getTransactionStates () const [inline]

6.178.2.11 Pointer<SessionState> activemq::state::ConnectionState::removeSession (const Pointer< SessionId > & *id*) [inline]

6.178.2.12 void activemq::state::ConnectionState::removeTempDestination (const Pointer< ActiveMQDestination > & *destination*) [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

6.178.2.13 Pointer<TransactionState> activemq::state::ConnectionState::removeTransactionState (const Pointer< TransactionId > & *id*) [inline]

6.178.2.14 void activemq::state::ConnectionState::reset (const Pointer< ConnectionInfo > & *info*)

6.178.2.15 void activemq::state::ConnectionState::shutdown ()

6.178.2.16 std::string activemq::state::ConnectionState::toString () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConnectionState.h**

6.179 activemq::state::ConnectionStateTracker Class Reference

`#include <src/main/activemq/state/ConnectionStateTracker.h>`Inheritance diagram for activemq::state::ConnectionStateTracker:

Public Member Functions

- **ConnectionStateTracker** ()
- virtual ~**ConnectionStateTracker** ()
- **Pointer< Tracked > track** (const **Pointer< Command >** &command) throw (decaf::io::IOException)
- void **trackBack** (const **Pointer< Command >** &command)
- void **restore** (const **Pointer< transport::Transport >** &transport) throw (decaf::io::IOException)
- virtual **Pointer< Command > processDestinationInfo** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveDestination** (**DestinationInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processProducerInfo** (**ProducerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveProducer** (**ProducerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processConsumerInfo** (**ConsumerInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveConsumer** (**ConsumerId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processSessionInfo** (**SessionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveSession** (**SessionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processConnectionInfo** (**ConnectionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRemoveConnection** (**ConnectionId** *id) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessage** (**Message** *message) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processMessageAck** (**MessageAck** *ack) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processBeginTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processPrepareTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionOnePhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processCommitTransactionTwoPhase** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- virtual **Pointer< Command > processRollbackTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)

- virtual **Pointer< Command > processEndTransaction** (**TransactionInfo** *info) throw (exceptions::ActiveMQException)
- bool **isRestoreConsumers** () const
- void **setRestoreConsumers** (bool restoreConsumers)
- bool **isRestoreProducers** () const
- void **setRestoreProducers** (bool restoreProducers)
- bool **isRestoreSessions** () const
- void **setRestoreSessions** (bool restoreSessions)
- bool **isTrackTransactions** () const
- void **setTrackTransactions** (bool trackTransactions)
- bool **isRestoreTransaction** () const
- void **setRestoreTransaction** (bool restoreTransaction)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool trackMessages)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int maxCacheSize)

Friends

- class **RemoveTransactionAction**

6.179.1 Constructor & Destructor Documentation

6.179.1.1 activemq::state::ConnectionStateTracker::ConnectionStateTracker ()

6.179.1.2 virtual activemq::state::ConnectionStateTracker::~ConnectionStateTracker () [virtual]

6.179.2 Member Function Documentation

6.179.2.1 int activemq::state::ConnectionStateTracker::getMaxCacheSize () const [inline]

6.179.2.2 bool activemq::state::ConnectionStateTracker::isRestoreConsumers () const [inline]

6.179.2.3 bool activemq::state::ConnectionStateTracker::isRestoreProducers () const [inline]

6.179.2.4 bool activemq::state::ConnectionStateTracker::isRestoreSessions () const [inline]

6.179.2.5 bool activemq::state::ConnectionStateTracker::isRestoreTransaction () const [inline]

6.179.2.6 bool activemq::state::ConnectionStateTracker::isTrackMessages () const [inline]

6.179.2.7 bool activemq::state::ConnectionStateTracker::isTrackTransactions () const [inline]

6.179.2.8 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processBeginTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 887).

6.179.2.9 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionOnePhase (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 887).

6.179.2.10 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processCommitTransactionTwoPhase (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 887).

6.179.2.11 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConnectionInfo (ConnectionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 888).

6.179.2.12 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processConsumerInfo (ConsumerInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 888).

6.179.2.13 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processDestinationInfo (DestinationInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 888).

6.179.2.14 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processEndTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 888).

6.179.2.15 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessage (Message * *message*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 889).

6.179.2.16 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processMessageAck (MessageAck * *ack*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 889).

6.179.2.17 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processPrepareTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 889).

6.179.2.18 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processProducerInfo (ProducerInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 890).

6.179.2.19 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConnection (ConnectionId * *id*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 890).

6.179.2.20 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveConsumer (ConsumerId * *id*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 890).

6.179.2.21 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveDestination (DestinationInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 890).

6.179.2.22 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveProducer (ProducerId * *id*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 890).

6.179.2.23 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRemoveSession (SessionId * *id*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 891).

6.179.2.24 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processRollbackTransaction (TransactionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 891).

6.179.2.25 virtual Pointer<Command> activemq::state::ConnectionStateTracker::processSessionInfo (SessionInfo * *info*) throw (exceptions::ActiveMQException) [virtual]

Implements **activemq::state::CommandVisitor** (p. 891).

6.179.2.26 void activemq::state::ConnectionStateTracker::restore (const Pointer< transport::Transport > & *transport*) throw (decaf::io::IOException)

6.179.2.27 void activemq::state::ConnectionStateTracker::setMaxCacheSize (int *maxCacheSize*) [inline]

6.179.2.28 void activemq::state::ConnectionStateTracker::setRestoreConsumers (bool *restoreConsumers*) [inline]

6.179.2.29 void activemq::state::ConnectionStateTracker::setRestoreProducers (bool *restoreProducers*) [inline]

6.179.2.30 void activemq::state::ConnectionStateTracker::setRestoreSessions (bool *restoreSessions*) [inline]

6.179.2.31 void activemq::state::ConnectionStateTracker::setRestoreTransaction (bool *restoreTransaction*) [inline]

6.179.2.32 void activemq::state::ConnectionStateTracker::setTrackMessages (bool *trackMessages*) [inline]

6.179.2.33 void activemq::state::ConnectionStateTracker::setTrackTransactions (bool *trackTransactions*) [inline]

6.179.2.34 Pointer<Tracked> activemq::state::ConnectionStateTracker::track (const Pointer< Command > & *command*) throw (decaf::io::IOException)

6.179.2.35 void activemq::state::ConnectionStateTracker::trackBack (const Pointer< Command > & *command*)

6.179.3 Friends And Related Function Documentation

6.179.3.1 friend class RemoveTransactionAction [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConnectionStateTracker.h**

6.180 activemq::commands::ConsumerControl Class Reference

`#include <src/main/activemq/commands/ConsumerControl.h>`Inheritance diagram for activemq::commands::ConsumerControl:

Public Member Functions

- **ConsumerControl** ()
- virtual ~**ConsumerControl** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConsumerControl** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual bool **isClose** () const
- virtual void **setClose** (bool **close**)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual int **getPrefetch** () const
- virtual void **setPrefetch** (int **prefetch**)
- virtual bool **isFlush** () const
- virtual void **setFlush** (bool **flush**)
- virtual bool **isStart** () const
- virtual void **setStart** (bool **start**)
- virtual bool **isStop** () const
- virtual void **setStop** (bool **stop**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERCONTROL** = 17

Protected Member Functions

- **ConsumerControl** (const **ConsumerControl** &)
- **ConsumerControl** & **operator=** (const **ConsumerControl** &)

Protected Attributes

- bool **close**
- **Pointer< ConsumerId >** **consumerId**
- int **prefetch**
- bool **flush**
- bool **start**
- bool **stop**

6.180.1 Constructor & Destructor Documentation

6.180.1.1 activemq::commands::ConsumerControl::ConsumerControl (const ConsumerControl &) [inline, protected]

6.180.1.2 activemq::commands::ConsumerControl::ConsumerControl ()

6.180.1.3 virtual activemq::commands::ConsumerControl::~ConsumerControl () [virtual]

6.180.2 Member Function Documentation

6.180.2.1 virtual ConsumerControl* activemq::commands::ConsumerControl::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.180.2.2 virtual void activemq::commands::ConsumerControl::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.180.2.3 virtual bool activemq::commands::ConsumerControl::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.180.2.4 virtual Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () [virtual]

6.180.2.5 virtual const Pointer<ConsumerId>& activemq::commands::ConsumerControl::getConsumerId () const [virtual]

6.180.2.6 virtual unsigned char activemq::commands::ConsumerControl::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.180.2.7 virtual int activemq::commands::ConsumerControl::getPrefetch () const [virtual]

6.180.2.8 virtual bool activemq::commands::ConsumerControl::isClose () const [virtual]

6.180.2.9 virtual bool activemq::commands::ConsumerControl::isFlush () const [virtual]

6.180.2.10 virtual bool activemq::commands::ConsumerControl::isStart () const [virtual]

6.180.2.11 virtual bool activemq::commands::ConsumerControl::isStop () const [virtual]

6.180.2.12 ConsumerControl& activemq::commands::ConsumerControl::operator= (const ConsumerControl &) [inline, protected]

6.180.2.13 virtual void activemq::commands::ConsumerControl::setClose (bool *close*) [virtual]

6.180.2.14 virtual void activemq::commands::ConsumerControl::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.180.2.15 virtual void activemq::commands::ConsumerControl::setFlush (bool *flush*) [virtual]

6.180.2.16 virtual void activemq::commands::ConsumerControl::setPrefetch (int *prefetch*) [virtual]

6.180.2.17 virtual void activemq::commands::ConsumerControl::setStart (bool *start*) [virtual]

6.180.2.18 virtual void activemq::commands::ConsumerControl::setStop (bool *stop*) [virtual]

6.180.2.19 virtual std::string activemq::commands::ConsumerControl::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.180.2.20 virtual Pointer<Command> activemq::commands::ConsumerControl::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.180.3 Field Documentation

6.180.3.1 bool activemq::commands::ConsumerControl::close [protected]

6.180.3.2 Pointer<ConsumerId> activemq::commands::ConsumerControl::consumerId [protected]

6.180.3.3 bool activemq::commands::ConsumerControl::flush [protected]

6.180.3.4 const unsigned char activemq::commands::ConsumerControl::ID_CONSUMERCONTROL = 17 [static]

6.180.3.5 int activemq::commands::ConsumerControl::prefetch [protected]

6.180.3.6 bool activemq::commands::ConsumerControl::start [protected]

6.180.3.7 bool activemq::commands::ConsumerControl::stop [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerControl.h**

6.181 activemq::wireformat::openwire::marshal::v3::ConsumerControlMa Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1033).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual ~**ConsumerControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.181.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1033). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.181.2 Constructor & Destructor Documentation

6.181.2.1 activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::ConsumerCont () [inline]

6.181.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::~ConsumerCo () [inline, virtual]

6.181.3 Member Function Documentation

6.181.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.181.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.181.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.181.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.181.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.181.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.181.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerControlMarshaller.h**

6.182 activemq::wireformat::openwire::marshal::v1::ConsumerControlMa Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1037).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual ~**ConsumerControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.182.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1037). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.182.2 Constructor & Destructor Documentation

6.182.2.1 activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::ConsumerCont () [inline]

6.182.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::~ConsumerCo () [inline, virtual]

6.182.3 Member Function Documentation

6.182.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.182.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.182.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.182.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.182.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.182.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.182.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerControlMarshaller.h**

6.183 activemq::wireformat::openwire::marshal::v2::ConsumerControlMa Class Reference

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1041).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerControlMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller:

Public Member Functions

- **ConsumerControlMarshaller** ()
- virtual ~**ConsumerControlMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.183.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerControlMarshaller** (p. 1041). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.183.2 Constructor & Destructor Documentation

6.183.2.1 activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::ConsumerCont () [inline]

6.183.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::~ConsumerCo () [inline, virtual]

6.183.3 Member Function Documentation

6.183.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.183.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.183.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.183.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.183.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.183.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.183.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerControlMarshaller.h**

6.184 activemq::commands::ConsumerId Class Reference

`#include <src/main/activemq/commands/ConsumerId.h>`Inheritance diagram for activemq::commands::ConsumerId:

Public Types

- typedef **decaf::lang::PointerComparator< ConsumerId > COMPARATOR**

Public Member Functions

- **ConsumerId** ()
- **ConsumerId** (const **ConsumerId** &other)
- **ConsumerId** (const **SessionId** &**sessionId**, long long consumerIdd)
- virtual ~**ConsumerId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConsumerId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- const **Pointer< SessionId >** & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &**connectionId**)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long **sessionId**)
- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual int **compareTo** (const **ConsumerId** &**value**) const
- virtual bool **equals** (const **ConsumerId** &**value**) const
- virtual bool **operator==** (const **ConsumerId** &**value**) const
- virtual bool **operator<** (const **ConsumerId** &**value**) const
- **ConsumerId** & **operator=** (const **ConsumerId** &other)

Static Public Attributes

- static const unsigned char **ID_CONSUMERID** = 122

Protected Attributes

- std::string **connectionId**
- long long **sessionId**
- long long **value**

6.184.1 Member Typedef Documentation

6.184.1.1 typedef decaf::lang::PointerComparator<ConsumerId> activemq::commands::ConsumerId::COMPARATOR

6.184.2 Constructor & Destructor Documentation

6.184.2.1 activemq::commands::ConsumerId::ConsumerId ()

6.184.2.2 activemq::commands::ConsumerId::ConsumerId (const ConsumerId & *other*)

6.184.2.3 activemq::commands::ConsumerId::ConsumerId (const SessionId & *sessionId*, long long *consumerIdd*) [inline]

References activemq::commands::SessionId::getConnectionId(), and activemq::commands::SessionId::getValue().

6.184.2.4 virtual activemq::commands::ConsumerId::~ConsumerId () [virtual]

6.184.3 Member Function Documentation

6.184.3.1 virtual ConsumerId* activemq::commands::ConsumerId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.184.3.2 virtual int activemq::commands::ConsumerId::compareTo (const ConsumerId & *value*) const [virtual]

6.184.3.3 virtual void activemq::commands::ConsumerId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.184.3.4 virtual bool activemq::commands::ConsumerId::equals (const ConsumerId & *value*) const [virtual]

6.184.3.5 virtual bool activemq::commands::ConsumerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.184.3.6 virtual std::string& activemq::commands::ConsumerId::getConnectionId () [virtual]

6.184.3.7 virtual const std::string& activemq::commands::ConsumerId::getConnectionId () const [virtual]

6.184.3.8 virtual unsigned char activemq::commands::ConsumerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.184.3.9 const Pointer<SessionId>& activemq::commands::ConsumerId::getParentId () const

6.184.3.10 virtual long long activemq::commands::ConsumerId::getSessionId () const [virtual]

6.184.3.11 virtual long long activemq::commands::ConsumerId::getValue () const [virtual]

6.184.3.12 virtual bool activemq::commands::ConsumerId::operator< (const ConsumerId & *value*) const [virtual]

6.184.3.13 ConsumerId& activemq::commands::ConsumerId::operator= (const ConsumerId & *other*)

6.184.3.14 virtual bool activemq::commands::ConsumerId::operator== (const ConsumerId & *value*) const [virtual]

6.184.3.15 virtual void activemq::commands::ConsumerId::setConnectionId (const std::string & *connectionId*) [virtual]

6.184.3.16 virtual void activemq::commands::ConsumerId::setSessionId (long long *sessionId*) [virtual]

6.184.3.17 virtual void activemq::commands::ConsumerId::setValue (long long *value*) [virtual]

6.184.3.18 virtual std::string activemq::commands::ConsumerId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.184.4 Field Documentation

6.184.4.1 std::string activemq::commands::ConsumerId::connectionId [protected]

6.184.4.2 const unsigned char activemq::commands::ConsumerId::ID_CONSUMERID = 122 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.184.4.3 long long activemq::commands::ConsumerId::sessionId [protected]

6.184.4.4 long long activemq::commands::ConsumerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerId.h**

6.185 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshal Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1050).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual ~**ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.185.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1050). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.185.2 Constructor & Destructor Documentation

6.185.2.1 activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::ConsumerIdMarsha () [inline]

6.185.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::~ConsumerIdMarsh () [inline, virtual]

6.185.3 Member Function Documentation

6.185.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.185.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.185.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.185.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.185.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.185.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.185.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerIdMarshaller.h**

6.186 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshal Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1054).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual ~**ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.186.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1054). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.186.2 Constructor & Destructor Documentation

6.186.2.1 activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::ConsumerIdMarshal () [inline]

6.186.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::~ConsumerIdMarsh () [inline, virtual]

6.186.3 Member Function Documentation

6.186.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.186.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.186.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.186.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.186.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.186.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.186.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerIdMarshaller.h**

6.187 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshal Class Reference

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1058).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller:

Public Member Functions

- **ConsumerIdMarshaller** ()
- virtual ~**ConsumerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.187.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerIdMarshaller** (p. 1058). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.187.2 Constructor & Destructor Documentation

6.187.2.1 activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::ConsumerIdMarshal () [inline]

6.187.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::~ConsumerIdMarsh () [inline, virtual]

6.187.3 Member Function Documentation

6.187.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.187.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.187.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.187.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.187.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.187.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.187.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerIdMarshaller.h**

6.188 activemq::commands::ConsumerInfo Class Reference

`#include <src/main/activemq/commands/ConsumerInfo.h>`Inheritance diagram for activemq::commands::ConsumerInfo:

Public Member Functions

- **ConsumerInfo** ()
- virtual ~**ConsumerInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ConsumerInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual bool **isBrowser** () const
- virtual void **setBrowser** (bool **browser**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual int **getPrefetchSize** () const
- virtual void **setPrefetchSize** (int **prefetchSize**)
- virtual int **getMaximumPendingMessageLimit** () const
- virtual void **setMaximumPendingMessageLimit** (int **maximumPendingMessageLimit**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool **dispatchAsync**)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &**selector**)
- virtual const std::string & **getSubscriptionName** () const
- virtual std::string & **getSubscriptionName** ()
- virtual void **setSubscriptionName** (const std::string &**subscriptionName**)
- virtual bool **isNoLocal** () const

- virtual void **setNoLocal** (bool **noLocal**)
- virtual bool **isExclusive** () const
- virtual void **setExclusive** (bool **exclusive**)
- virtual bool **isRetroactive** () const
- virtual void **setRetroactive** (bool **retroactive**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual const **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** () const
- virtual **Pointer**< **BooleanExpression** > & **getAdditionalPredicate** ()
- virtual void **setAdditionalPredicate** (const **Pointer**< **BooleanExpression** > &**additionalPredicate**)
- virtual bool **isNetworkSubscription** () const
- virtual void **setNetworkSubscription** (bool **networkSubscription**)
- virtual bool **isOptimizedAcknowledge** () const
- virtual void **setOptimizedAcknowledge** (bool **optimizedAcknowledge**)
- virtual bool **isNoRangeAcks** () const
- virtual void **setNoRangeAcks** (bool **noRangeAcks**)
- virtual bool **isConsumerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONSUMERINFO** = 5

Protected Member Functions

- **ConsumerInfo** (const **ConsumerInfo** &)
- **ConsumerInfo** & **operator**= (const **ConsumerInfo** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- bool **browser**
- **Pointer**< **ActiveMQDestination** > **destination**
- int **prefetchSize**
- int **maximumPendingMessageLimit**
- bool **dispatchAsync**
- std::string **selector**
- std::string **subscriptionName**
- bool **noLocal**

- bool **exclusive**
- bool **retroactive**
- unsigned char **priority**
- std::vector< **decaf::lang::Pointer**< **BrokerId** > > **brokerPath**
- **Pointer**< **BooleanExpression** > **additionalPredicate**
- bool **networkSubscription**
- bool **optimizedAcknowledge**
- bool **noRangeAcks**

6.188.1 Constructor & Destructor Documentation

6.188.1.1 activemq::commands::ConsumerInfo::ConsumerInfo (const ConsumerInfo &) [inline, protected]

6.188.1.2 activemq::commands::ConsumerInfo::ConsumerInfo ()

6.188.1.3 virtual activemq::commands::ConsumerInfo::~ConsumerInfo () [virtual]

6.188.2 Member Function Documentation

6.188.2.1 virtual ConsumerInfo* activemq::commands::ConsumerInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.188.2.2 virtual void activemq::commands::ConsumerInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.188.2.3 virtual bool activemq::commands::ConsumerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.188.2.4 virtual Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () [virtual]

6.188.2.5 virtual const Pointer<BooleanExpression>& activemq::commands::ConsumerInfo::getAdditionalPredicate () const [virtual]

6.188.2.6 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () [virtual]

6.188.2.7 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ConsumerInfo::getBrokerPath () const [virtual]

6.188.2.8 virtual Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () [virtual]

6.188.2.9 virtual const Pointer<ConsumerId>& activemq::commands::ConsumerInfo::getConsumerId () const [virtual]

6.188.2.10 virtual unsigned char activemq::commands::ConsumerInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.188.2.11 virtual Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () [virtual]

6.188.2.12 virtual const Pointer<ActiveMQDestination>& activemq::commands::ConsumerInfo::getDestination () const [virtual]

6.188.2.13 virtual int activemq::commands::ConsumerInfo::getMaximumPendingMessageLimit () const [virtual]

6.188.2.14 virtual int activemq::commands::ConsumerInfo::getPrefetchSize () const [virtual]

6.188.2.15 virtual unsigned char activemq::commands::ConsumerInfo::getPriority () const [virtual]

6.188.2.16 virtual std::string& activemq::commands::ConsumerInfo::getSelector () [virtual]

6.188.2.17 virtual const std::string& activemq::commands::ConsumerInfo::getSelector () const [virtual]

6.188.2.18 virtual std::string& activemq::commands::ConsumerInfo::getSubscriptionName () [virtual]

6.188.2.19 virtual const std::string& activemq::commands::ConsumerInfo::getSubscriptionName () const [virtual]

6.188.2.20 virtual bool activemq::commands::ConsumerInfo::isBrowser () const [virtual]

6.188.2.21 virtual bool activemq::commands::ConsumerInfo::isConsumerInfo () const [inline, virtual]

Returns:

an answer of true to the **isConsumerInfo()** (p. 1066) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.188.2.22 virtual bool activemq::commands::ConsumerInfo::isDispatchAsync () const [virtual]

6.188.2.23 virtual bool activemq::commands::ConsumerInfo::isExclusive () const [virtual]

6.188.2.24 virtual bool activemq::commands::ConsumerInfo::isNetworkSubscription () const [virtual]

6.188.2.25 virtual bool activemq::commands::ConsumerInfo::isNoLocal () const [virtual]

6.188.2.26 virtual bool activemq::commands::ConsumerInfo::isNoRangeAcks () const [virtual]

6.188.2.27 virtual bool activemq::commands::ConsumerInfo::isOptimizedAcknowledge () const [virtual]

6.188.2.28 virtual bool activemq::commands::ConsumerInfo::isRetroactive () const [virtual]

6.188.2.29 ConsumerInfo& activemq::commands::ConsumerInfo::operator= (const ConsumerInfo &) [inline, protected]

6.188.2.30 virtual void activemq::commands::ConsumerInfo::setAdditionalPredicate (const Pointer< BooleanExpression > & *additionalPredicate*) [virtual]

6.188.2.31 virtual void activemq::commands::ConsumerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.188.2.32 virtual void activemq::commands::ConsumerInfo::setBrowser (bool *browser*) [virtual]

6.188.2.33 virtual void activemq::commands::ConsumerInfo::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.188.2.34 virtual void activemq::commands::ConsumerInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.188.2.35 virtual void activemq::commands::ConsumerInfo::setDispatchAsync (bool *dispatchAsync*) [virtual]

6.188.2.36 virtual void activemq::commands::ConsumerInfo::setExclusive (bool *exclusive*) [virtual]

6.188.2.37 virtual void activemq::commands::ConsumerInfo::setMaximumPendingMessageLimit (int *maximumPendingMessageLimit*) [virtual]

6.188.2.38 virtual void activemq::commands::ConsumerInfo::setNetworkSubscription (bool *networkSubscription*) [virtual]

6.188.2.39 virtual void activemq::commands::ConsumerInfo::setNoLocal (bool *noLocal*) [virtual]

6.188.2.40 virtual void activemq::commands::ConsumerInfo::setNoRangeAcks (bool *noRangeAcks*) [virtual]

6.188.2.41 virtual void ac-

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.188.2.48 virtual Pointer<Command> activemq::commands::ConsumerInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.188.3 Field Documentation

6.188.3.1 Pointer<BooleanExpression> activemq::commands::ConsumerInfo::additionalPredicate [protected]

6.188.3.2 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ConsumerInfo::brokerPath [protected]

6.188.3.3 bool activemq::commands::ConsumerInfo::browser [protected]

6.188.3.4 Pointer<ConsumerId> activemq::commands::ConsumerInfo::consumerId [protected]

6.188.3.5 Pointer<ActiveMQDestination> activemq::commands::ConsumerInfo::destination [protected]

6.188.3.6 bool activemq::commands::ConsumerInfo::dispatchAsync [protected]

6.188.3.7 bool activemq::commands::ConsumerInfo::exclusive [protected]

6.188.3.8 const unsigned char activemq::commands::ConsumerInfo::ID_CONSUMERINFO = 5 [static]

6.188.3.9 int activemq::commands::ConsumerInfo::maximumPendingMessageLimit [protected]

6.188.3.10 bool activemq::commands::ConsumerInfo::networkSubscription [protected]

6.188.3.11 bool activemq::commands::ConsumerInfo::noLocal [protected]

6.188.3.12 bool activemq::commands::ConsumerInfo::noRangeAcks [protected]

6.188.3.13 bool activemq::commands::ConsumerInfo::optimizedAcknowledge [protected]

6.188.3.14 int activemq::commands::ConsumerInfo::prefetchSize [protected]

6.188.3.15 unsigned char activemq::commands::ConsumerInfo::priority [protected]

6.188.3.16 bool activemq::commands::ConsumerInfo::retroactive [protected]

6.188.3.17 std::string activemq::commands::ConsumerInfo::selector [protected]

6.188.3.18 std::string activemq::commands::ConsumerInfo::subscriptionName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ConsumerInfo.h**

6.189 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1070).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ConsumerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual ~**ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.189.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1070). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.189.2 Constructor & Destructor Documentation

6.189.2.1 activemq::wireformat::openwire::marshal::v3::ConsumerInfoMar () [inline]

6.189.2.2 virtual activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::~ConsumerInfoMa () [inline, virtual]

6.189.3 Member Function Documentation

6.189.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.189.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.189.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.189.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.189.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.189.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.189.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ConsumerInfoMarshaller.h**

6.190 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1074).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ConsumerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual ~**ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.190.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1074). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.190.2 Constructor & Destructor Documentation

6.190.2.1 activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::ConsumerInfoMar () [inline]

6.190.2.2 virtual activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::~ConsumerInfoMa () [inline, virtual]

6.190.3 Member Function Documentation

6.190.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.190.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.190.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.190.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.190.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.190.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.190.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ConsumerInfoMarshaller.h**

6.191 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1078).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ConsumerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller:

Public Member Functions

- **ConsumerInfoMarshaller** ()
- virtual ~**ConsumerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.191.1 Detailed Description

Marshaling code for Open Wire Format for **ConsumerInfoMarshaller** (p. 1078). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.191.2 Constructor & Destructor Documentation

6.191.2.1 activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::ConsumerInfoMar () [inline]

6.191.2.2 virtual activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::~ConsumerInfoMa () [inline, virtual]

6.191.3 Member Function Documentation

6.191.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.191.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.191.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.191.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.191.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.191.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.191.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ConsumerInfoMarshaller.h**

6.192 activemq::state::ConsumerState Class Reference

`#include <src/main/activemq/state/ConsumerState.h>`

Public Member Functions

- **ConsumerState** (const **Pointer**< **ConsumerInfo** > &info)
- virtual ~**ConsumerState** ()
- std::string **toString** () const
- const **Pointer**< **ConsumerInfo** > & **getInfo** () const

6.192.1 Constructor & Destructor Documentation

6.192.1.1 activemq::state::ConsumerState::ConsumerState (const Pointer< ConsumerInfo > & *info*)

6.192.1.2 virtual activemq::state::ConsumerState::~ConsumerState () [virtual]

6.192.2 Member Function Documentation

6.192.2.1 const Pointer<ConsumerInfo>& activemq::state::ConsumerState::getInfo () const [inline]

6.192.2.2 std::string activemq::state::ConsumerState::toString () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ConsumerState.h**

6.193 activemq::commands::ControlCommand Class Reference

`#include <src/main/activemq/commands/ControlCommand.h>`Inheritance diagram for activemq::commands::ControlCommand:

Public Member Functions

- **ControlCommand** ()
- virtual ~**ControlCommand** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ControlCommand** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getCommand** () const
- virtual std::string & **getCommand** ()
- virtual void **setCommand** (const std::string &**command**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_CONTROLCOMMAND** = 14

Protected Member Functions

- **ControlCommand** (const **ControlCommand** &)
- **ControlCommand** & **operator**= (const **ControlCommand** &)

Protected Attributes

- std::string **command**

6.193.1 Constructor & Destructor Documentation

6.193.1.1 activemq::commands::ControlCommand::ControlCommand (const ControlCommand &) [inline, protected]

6.193.1.2 activemq::commands::ControlCommand::ControlCommand ()

6.193.1.3 virtual activemq::commands::ControlCommand::~ControlCommand () [virtual]

6.193.2 Member Function Documentation

6.193.2.1 virtual ControlCommand* activemq::commands::ControlCommand::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.193.2.2 virtual void activemq::commands::ControlCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.193.2.3 virtual bool activemq::commands::ControlCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.193.2.4 virtual std::string& activemq::commands::ControlCommand::getCommand () [virtual]

6.193.2.5 virtual const std::string& activemq::commands::ControlCommand::getCommand () const [virtual]

6.193.2.6 virtual unsigned char activemq::commands::ControlCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.193.2.7 ControlCommand& activemq::commands::ControlCommand::operator= (const ControlCommand &) [inline, protected]

6.193.2.8 virtual void activemq::commands::ControlCommand::setCommand (const std::string & *command*) [virtual]

6.193.2.9 virtual std::string activemq::commands::ControlCommand::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.193.2.10 virtual Pointer<Command> activemq::commands::ControlCommand::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.193.3 Field Documentation

6.193.3.1 std::string activemq::commands::ControlCommand::command [protected]

6.193.3.2 const unsigned char activemq::commands::ControlCommand::ID_CONTROLCOMMAND = 14 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ControlCommand.h**

6.194 activemq::wireformat::openwire::marshal::v1::ControlCommandMa Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1087).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ControlCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual ~**ControlCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.194.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1087). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.194.2 Constructor & Destructor Documentation

6.194.2.1 activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::ControlComma () [inline]

6.194.2.2 virtual activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::~ControlComm () [inline, virtual]

6.194.3 Member Function Documentation

6.194.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.194.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::getDataStructur () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.194.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.194.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.194.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.194.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.194.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ControlCommandMarshaller.h**

6.195 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1091).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ControlCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual ~**ControlCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.195.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1091). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.195.2 Constructor & Destructor Documentation

6.195.2.1 activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::ControlComma () [inline]

6.195.2.2 virtual activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::~ControlComm () [inline, virtual]

6.195.3 Member Function Documentation

6.195.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.195.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::getDataStructur () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.195.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.195.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.195.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.195.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.195.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ControlCommandMarshaller.h**

6.196 activemq::wireformat::openwire::marshal::v2::ControlCommandMa Class Reference

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1095).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller:

Public Member Functions

- **ControlCommandMarshaller** ()
- virtual ~**ControlCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.196.1 Detailed Description

Marshaling code for Open Wire Format for **ControlCommandMarshaller** (p. 1095). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.196.2 Constructor & Destructor Documentation

6.196.2.1 activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::ControlComma () [inline]

6.196.2.2 virtual activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::~ControlComm () [inline, virtual]

6.196.3 Member Function Documentation

6.196.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.196.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::getDataStructur () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.196.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.196.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.196.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.196.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.196.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ControlCommandMarshaller.h**

6.197 decaf::util::concurrent::CountDownLatch Class Reference

`#include <src/main/decaf/util/concurrent/CountDownLatch.h>`

Public Member Functions

- **CountDownLatch** (int count)
  Constructor.
- virtual ~**CountDownLatch** ()
- virtual void **await** () throw (lang::Exception)
  Waits for the Count to be zero, and then returns.
- virtual bool **await** (unsigned long timeOut) throw (lang::Exception)
  Waits for the Count to hit zero, or a timeout.
- virtual void **countDown** ()
  Counts down the latch, releasing all waiting threads when the count hits zero.
- virtual int **getCount** () const
  Gets the current count.

6.197.1 Constructor & Destructor Documentation

6.197.1.1 decaf::util::concurrent::CountDownLatch::CountDownLatch (int *count*)

Constructor.

Parameters:

count - number to count down from.

6.197.1.2 virtual decaf::util::concurrent::CountDownLatch::~CountDownLatch () [virtual]

6.197.2 Member Function Documentation

6.197.2.1 virtual bool decaf::util::concurrent::CountDownLatch::await (unsigned long *timeOut*) throw (lang::Exception) [virtual]

Waits for the Count to hit zero, or a timeout.

Parameters:

timeOut - time in milliseconds to wait.

Returns:

true if the wait made it to count zero, otherwise false

6.197.2.2 virtual void decaf::util::concurrent::CountDownLatch::await () throw (lang::Exception) [virtual]

Waits for the Count to be zero, and then returns.

Exceptions:

CMSException

6.197.2.3 virtual void decaf::util::concurrent::CountDownLatch::countDown () [virtual]

Counts down the latch, releasing all waiting threads when the count hits zero.

6.197.2.4 virtual int decaf::util::concurrent::CountDownLatch::getCount () const [inline, virtual]

Gets the current count.

Returns:

int count value

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**CountDownLatch.h**

6.198 activemq::commands::DataArrayResponse Class Reference

`#include <src/main/activemq/commands/DataArrayResponse.h>`Inheritance diagram for activemq::commands::DataArrayResponse:

Public Member Functions

- **DataArrayResponse** ()
- virtual ~**DataArrayResponse** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **DataArrayResponse** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** () const
- virtual std::vector< **decaf::lang::Pointer**< **DataStructure** > > & **getData** ()
- virtual void **setData** (const std::vector< **decaf::lang::Pointer**< **DataStructure** > > &**data**)

Static Public Attributes

- static const unsigned char **ID_DATAARRAYRESPONSE** = 33

Protected Member Functions

- **DataArrayResponse** (const **DataArrayResponse** &)
- **DataArrayResponse** & **operator**= (const **DataArrayResponse** &)

Protected Attributes

- std::vector< **decaf::lang::Pointer**< **DataStructure** > > **data**

6.198.1 Constructor & Destructor Documentation

6.198.1.1 activemq::commands::DataArrayResponse::DataArrayResponse (const DataArrayResponse &) [inline, protected]

6.198.1.2 activemq::commands::DataArrayResponse::DataArrayResponse ()

6.198.1.3 virtual activemq::commands::DataArrayResponse::~DataArrayResponse () [virtual]

6.198.2 Member Function Documentation

6.198.2.1 virtual DataArrayResponse* activemq::commands::DataArrayResponse::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.198.2.2 virtual void activemq::commands::DataArrayResponse::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2232).

6.198.2.3 virtual bool activemq::commands::DataArrayResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.198.2.4 virtual std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () [virtual]

6.198.2.5 virtual const std::vector< decaf::lang::Pointer<DataStructure> >& activemq::commands::DataArrayResponse::getData () const [virtual]

6.198.2.6 virtual unsigned char activemq::commands::DataArrayResponse::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.198.2.7 DataArrayResponse& activemq::commands::DataArrayResponse::operator= (const DataArrayResponse &) [inline, protected]

Reimplemented from **activemq::commands::Response** (p. 2233).

6.198.2.8 virtual void activemq::commands::DataArrayResponse::setData (const std::vector< decaf::lang::Pointer< DataStructure > > & *data*) [virtual]

6.198.2.9 virtual std::string activemq::commands::DataArrayResponse::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.198.3 Field Documentation

6.198.3.1 std::vector< decaf::lang::Pointer<DataStructure> > activemq::commands::DataArrayResponse::data [protected]

6.198.3.2 const unsigned char activemq::commands::DataArrayResponse::ID_DATAARRAYRESPONSE = 33 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataArrayResponse.h**

6.199 activemq::wireformat::openwire::marshal::v3::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1104).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual ~**DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.199.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1104). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.199.2 Constructor & Destructor Documentation

6.199.2.1 activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::DataArrayR () [inline]

6.199.2.2 virtual activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::~DataArray () [inline, virtual]

6.199.3 Member Function Documentation

6.199.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.199.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.199.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.199.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.199.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.199.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2249).

6.199.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataArrayResponseMarshaller.h**

6.200 activemq::wireformat::openwire::marshal::v1::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1108).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual ~**DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.200.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1108). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.200.2 Constructor & Destructor Documentation

6.200.2.1 activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::DataArrayR () [inline]

6.200.2.2 virtual activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::~DataArray () [inline, virtual]

6.200.3 Member Function Documentation

6.200.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.200.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.200.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.200.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.200.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.200.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2254).

6.200.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2255).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataArrayResponseMarshaller.h**

6.201 activemq::wireformat::openwire::marshal::v2::DataArrayResponse Class Reference

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1112).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/DataArrayResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller:

Public Member Functions

- **DataArrayResponseMarshaller** ()
- virtual ~**DataArrayResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.201.1 Detailed Description

Marshaling code for Open Wire Format for **DataArrayResponseMarshaller** (p. 1112). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.201.2 Constructor & Destructor Documentation

6.201.2.1 activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::DataArrayR () [inline]

6.201.2.2 virtual activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::~DataArray () [inline, virtual]

6.201.3 Member Function Documentation

6.201.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.201.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.201.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.201.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.201.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.201.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2244).

6.201.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataArrayResponseMarshaller.h**

6.202 decaf::io::DataInputStream Class Reference

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

`#include <src/main/decaf/io/DataInputStream.h>`Inheritance diagram for decaf::io::DataInputStream:

Public Member Functions

- **DataInputStream** (**InputStream** ∗**inputStream**, bool **own**=false)

  *Creates a **DataInputStream** (p. 1116) that uses the specified underlying **InputStream** (p. 1406).*

- virtual ∼**DataInputStream** ()
- virtual int **read** (std::vector< unsigned char > &buffer) throw (io::IOException)

  Reads some number of bytes from the contained input stream and stores them into the buffer array b.

- virtual int **read** (unsigned char ∗buffer, std::size_t offset, std::size_t length) throw (io::IOException, lang::exceptions::NullPointerException)

  Reads up to len bytes of data from the contained input stream into an array of bytes.

- virtual bool **readBoolean** () throw (io::IOException, io::EOFException)

  Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.

- virtual char **readByte** () throw (io::IOException, io::EOFException)

  Reads and returns one input byte.

- virtual unsigned char **readUnsignedByte** () throw (io::IOException, io::EOFException)

  Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

- virtual char **readChar** () throw (io::IOException, io::EOFException)

  Reads an input char and returns the char value.

- virtual double **readDouble** () throw (io::IOException, io::EOFException)

  Reads eight input bytes and returns a double value.

- virtual float **readFloat** () throw (io::IOException, io::EOFException)

  Reads four input bytes and returns a float value.

- virtual int **readInt** () throw (io::IOException, io::EOFException)

  Reads four input bytes and returns an int value.

- virtual long long **readLong** () throw (io::IOException, io::EOFException)

  Reads eight input bytes and returns a long value.

- virtual short **readShort** () throw (io::IOException, io::EOFException)

  Reads two input bytes and returns a short value.

- virtual unsigned short **readUnsignedShort** () throw (io::IOException, io::EOFException)

  Reads two input bytes and returns an int value in the range 0 through 65535.

- virtual std::string **readString** () throw (io::IOException, io::EOFException)

  Reads an null terminated ASCII string to the stream and returns the string to the caller.

- virtual std::string **readUTF** () throw (io::IOException, io::EOFException, io::UTFDataFormatException)

  Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException.

- virtual void **readFully** (std::vector< unsigned char > &buffer) throw (io::IOException, io::EOFException)

  Reads some bytes from an input stream and stores them into the buffer array buffer.

- virtual void **readFully** (unsigned char *buffer, std::size_t offset, std::size_t length) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException)

  Reads length bytes from an input stream.

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes.

6.202.1 Detailed Description

A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way. An application uses a data output stream to write data that can later be read by a data input stream.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped `InputStream (p. 1406)` is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataInputStream (p. 1116) os = new **DataInputStream** (p. 1116)(new InputStream(), true)

Since:

1.0

6.202.2 Constructor & Destructor Documentation

6.202.2.1 decaf::io::DataInputStream::DataInputStream (InputStream * *inputStream*, bool *own* = `false`)

Creates a **DataInputStream** (p. 1116) that uses the specified underlying **InputStream** (p. 1406).

Parameters:

inputStream the **InputStream** (p. 1406) instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.202.2.2 virtual decaf::io::DataInputStream::~DataInputStream () [virtual]

6.202.3 Member Function Documentation

6.202.3.1 virtual int decaf::io::DataInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]

Reads up to len bytes of data from the contained input stream into an array of bytes. An attempt is made to read as many as len bytes, but a smaller number may be read, possibly zero. The number of bytes actually read is returned as an integer.

This method blocks until input data is available, end of file is detected, or an exception is thrown.

If buffer is null, a NullPointerException is thrown.

If off is negative, or len is negative then an IndexOutOfBoundsException is thrown, if off + len is greater that the allocated length of the array, an **IOException** (p. 1477) will result depending on the platform and compiler settings.

If len is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into buffer.

The first byte read is stored into element b[off], the next one into buffer[off+1], and so on. The number of bytes read is, at most, equal to len. Let k be the number of bytes actually read; these bytes will be stored in elements buffer[off] through buffer[off+k-1], leaving elements buffer[off+k] through buffer[off+len-1] unaffected.

In every case, elements buffer[0] through buffer[off] and elements buffer[off+len] through buffer[buffer.length-1] are unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p. 1477) is thrown. In particular, an **IOException** (p. 1477) is thrown if the input stream has been closed.

Parameters:

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Returns:

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions:

IOException *(p. 1477)*

NullPointerException if the buffer is null

Reimplemented from **decaf::io::FilterInputStream** (p. 1319).

6.202.3.2 virtual int decaf::io::DataInputStream::read (std::vector< unsigned char > & *buffer*) throw (io::IOException) [virtual]

Reads some number of bytes from the contained input stream and stores them into the buffer array b. The number of bytes actually read is returned as an integer. This method blocks until input data is available, end of file is detected, or an exception is thrown.

If the length of buffer is zero, then no bytes are read and 0 is returned; otherwise, there is an attempt to read at least one byte. If no byte is available because the stream is at end of file, the value -1 is returned; otherwise, at least one byte is read and stored into buffer.

The first byte read is stored into element buffer[0], the next one into buffer[1], and so on. The number of bytes read is, at most, equal to the length of buffer. Let k be the number of bytes actually read; these bytes will be stored in elements b[0] through b[k-1], leaving elements buffer[k] through buffer[buffer.length-1] unaffected.

If the first byte cannot be read for any reason other than end of file, then an **IOException** (p. 1477) is thrown. In particular, an **IOException** (p. 1477) is thrown if the input stream has been closed.

The read(buffer) method has the same effect as: read(buffer, 0, b.length)

Parameters:

buffer - byte array to insert read data into

Returns:

the total number of bytes read, or -1 if there is no more data because the stream is EOF.

Exceptions:

IOException *(p. 1477)*

6.202.3.3 virtual bool decaf::io::DataInputStream::readBoolean () throw (io::IOException, io::EOFException) [virtual]

Reads one input byte and returns true if that byte is nonzero, false if that byte is zero.

Returns:

the boolean value read.

Exceptions:

IOException *(p. 1477)*
EOFException *(p. 1265)*

6.202.3.4 virtual char decaf::io::DataInputStream::readByte () throw (io::IOException, io::EOFException) [virtual]

Reads and returns one input byte. The byte is treated as a signed value in the range -128 through 127, inclusive.

Returns:

the 8-bit value read.

Exceptions:

IOException *(*p. *1477)*

EOFException *(*p. *1265)*

6.202.3.5 virtual char decaf::io::DataInputStream::readChar () throw (io::IOException, io::EOFException) [virtual]

Reads an input char and returns the char value. A ascii char is made up of one bytes. This returns the same result as `readByte`

Returns:

the 8 bit char read

Exceptions:

IOException *(*p. *1477)*

EOFException *(*p. *1265)*

6.202.3.6 virtual double decaf::io::DataInputStream::readDouble () throw (io::IOException, io::EOFException) [virtual]

Reads eight input bytes and returns a double value. It does this by first constructing a long long value in exactly the manner of the readlong method, then converting this long value to a double in exactly the manner of the method Double.longBitsToDouble.

Returns:

the double value read

Exceptions:

IOException *(*p. *1477)*

EOFException *(*p. *1265)*

6.202.3.7 virtual float decaf::io::DataInputStream::readFloat () throw (io::IOException, io::EOFException) [virtual]

Reads four input bytes and returns a float value. It does this by first constructing an int value in exactly the manner of the readInt method, then converting this int value to a float in exactly the manner of the method Float.intBitsToFloat.

Returns:

the float value read

Exceptions:

IOException *(*p. *1477)*

EOFException *(*p. *1265)*

6.202.3.8 virtual void decaf::io::DataInputStream::readFully (unsigned char * *buffer*, std::size_t *offset*, std::size_t *length*) throw (io::IOException, io::EOFException, lang::exceptions::NullPointerException) [virtual]

Reads length bytes from an input stream. This method blocks until one of the following conditions occurs: * length bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1265) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1477) other than **EOFException** (p. 1265) is thrown.

If buffer is null, a NullPointerException is thrown. If offset is negative, or len is negative, or offset+length is greater than the length of the array buffer, then an IndexOutOfBoundsException is thrown. If len is zero, then no bytes are read. Otherwise, the first byte read is stored into element buffer[off], the next one into buffer[offset+1], and so on. The number of bytes read is, at most, equal to len.

Parameters:

buffer - byte array to insert read data into

offset - location in buffer to start writing

length - number of bytes to read

Exceptions:

IOException *(*p. ***1477****)*

EOFException *(*p. ***1265****)*

NullPointerException if the buffer is null

6.202.3.9 virtual void decaf::io::DataInputStream::readFully (std::vector< unsigned char > & *buffer*) throw (io::IOException, io::EOFException) [virtual]

Reads some bytes from an input stream and stores them into the buffer array buffer. The number of bytes read is equal to the length of buffer.

This method blocks until one of the following conditions occurs: * buffer.size() bytes of input data are available, in which case a normal return is made. * End of file is detected, in which case an **EOFException** (p. 1265) is thrown. * An I/O error occurs, in which case an **IOException** (p. 1477) other than **EOFException** (p. 1265) is thrown.

If buffer.size() is zero, then no bytes are read. Otherwise, the first byte read is stored into element b[0], the next one into buffer[1], and so on. If an exception is thrown from this method, then it may be that some but not all bytes of buffer have been updated with data from the input stream.

Parameters:

buffer - vector of char that is read to its size()

Exceptions:

IOException *(*p. ***1477****)*

EOFException *(*p. ***1265****)*

6.202.3.10 virtual int decaf::io::DataInputStream::readInt () throw (io::IOException, io::EOFException) [virtual]

Reads four input bytes and returns an int value. Let a be the first byte read, b be the second byte, c be the third byte, and d be the fourth byte. The value returned is:

(((a & 0xff) << 24) | ((b & 0xff) << 16) | ((c & 0xff) << 8) | (d & 0xff))

Returns:

the int value read

Exceptions:

IOException (p. ***1477***)

EOFException (p. ***1265***)

6.202.3.11 virtual long long decaf::io::DataInputStream::readLong () throw (io::IOException, io::EOFException) [virtual]

Reads eight input bytes and returns a long value. Let a be the first byte read, b be the second byte, c be the third byte, d be the fourth byte, e be the fifth byte, f be the sixth byte, g be the seventh byte, and h be the eighth byte. The value returned is: (((long)(a & 0xff) << 56) | ((long)(b & 0xff) << 48) | ((long)(c & 0xff) << 40) | ((long)(d & 0xff) << 32) | ((long)(e & 0xff) << 24) | ((long)(f & 0xff) << 16) | ((long)(g & 0xff) << 8) | ((long)(h & 0xff)))

Returns:

the 64 bit long long read

Exceptions:

IOException (p. ***1477***)

EOFException (p. ***1265***)

6.202.3.12 virtual short decaf::io::DataInputStream::readShort () throw (io::IOException, io::EOFException) [virtual]

Reads two input bytes and returns a short value. Let a be the first byte read and b be the second byte. The value returned is: (short)((a << 8) | (b & 0xff))

Returns:

the 16 bit short value read

Exceptions:

IOException (p. ***1477***)

EOFException (p. ***1265***)

6.202.3.13 virtual std::string decaf::io::DataInputStream::readString () throw (io::IOException, io::EOFException) [virtual]

Reads an null terminated ASCII string to the stream and returns the string to the caller.

Returns:

string object containing the string read.

Exceptions:

IOException (p. ***1477***)
EOFException (p. ***1265***)

6.202.3.14 virtual unsigned char decaf::io::DataInputStream::readUnsignedByte () throw (io::IOException, io::EOFException) [virtual]

Reads one input byte, zero-extends it to type int, and returns the result, which is therefore in the range 0 through 255.

Returns:

the 8 bit unsigned value read

Exceptions:

IOException (p. ***1477***)
EOFException (p. ***1265***)

6.202.3.15 virtual unsigned short decaf::io::DataInputStream::readUnsignedShort () throw (io::IOException, io::EOFException) [virtual]

Reads two input bytes and returns an int value in the range 0 through 65535. Let a be the first byte read and b be the second byte. The value returned is: (((a & 0xff) << 8) | (b & 0xff))

Returns:

the 16 bit unsigned short read

Exceptions:

IOException (p. ***1477***)
EOFException (p. ***1265***)

6.202.3.16 virtual std::string decaf::io::DataInputStream::readUTF () throw (io::IOException, io::EOFException, io::UTFDataFormatException) [virtual]

Reads a modified UTF-8 encoded string in ASCII format and returns it, this is only useful if you know for sure that the string that is to be read was a string that contained all ASCII values (0-255), if so this method will throw a UTFFormatException. This method reads String value written from a Java **DataOutputStream** (p. 1125) and assumes that the length prefix the precedes the encoded UTF-8 bytes is an unsigned short, which implies that the String will be no longer than 65535 characters.

Returns:

The decoded string read from stream.

Exceptions:

IOException *(*p. *1477)*

EOFException *(*p. *1265)*

UTFDataFormatException *(*p. *2683)*

6.202.3.17 virtual std::size_t decaf::io::DataInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Makes an attempt to skip over n bytes of data from the input stream, discarding the skipped bytes. However, it may skip over some smaller number of bytes, possibly zero. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. This method never throws an **EOFException** (p. 1265). The actual number of bytes skipped is returned.

Parameters:

num - number of bytes to skip

Returns:

the total number of bytes skipped

Reimplemented from **decaf::io::FilterInputStream** (p. 1320).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataInputStream.h**

6.203 decaf::io::DataOutputStream Class Reference

A data output stream lets an application write primitive Java data types to an output stream in a portable way.

`#include <src/main/decaf/io/DataOutputStream.h>`Inheritance diagram for decaf::io::DataOutputStream:

Public Member Functions

- **DataOutputStream** (**OutputStream** ∗**outputStream**, bool **own**=false)

  Creates a new data output stream to write data to the specified underlying output stream.

- virtual ∼**DataOutputStream** ()
- virtual std::size_t **size** () const

  Returns the current value of the counter written, the number of bytes written to this data output stream so far.

- virtual void **write** (unsigned char c) throw (IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const unsigned char ∗**buffer**, std::size_t offset, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const std::vector< unsigned char > &**buffer**) throw (IOException)

  Writes an array of bytes to the output stream.

- virtual void **writeBoolean** (bool value) throw (IOException)

  Writes a boolean to the underlying output stream as a 1-byte value.

- virtual void **writeByte** (unsigned char value) throw (IOException)

  Writes out a byte to the underlying output stream as a 1-byte value.

- virtual void **writeShort** (short value) throw (IOException)

  Writes a short to the underlying output stream as two bytes, high byte first.

- virtual void **writeUnsignedShort** (unsigned short value) throw (IOException)

  Writes a unsigned short to the bytes message stream as a 2 byte value.

- virtual void **writeChar** (char value) throw (IOException)

  Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.

- virtual void **writeInt** (int value) throw (IOException)

  Writes an int to the underlying output stream as four bytes, high byte first.

- virtual void **writeLong** (long long value) throw (IOException)

  Writes an 64 bit long to the underlying output stream as eight bytes, high byte first.

- virtual void **writeFloat** (float value) throw (IOException)

  Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first.

- virtual void **writeDouble** (double value) throw (IOException)

  Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first.

- virtual void **writeBytes** (const std::string &value) throw (IOException)

  Writes out the string to the underlying output stream as a sequence of bytes.

- virtual void **writeChars** (const std::string &value) throw (IOException)

  Writes a string to the underlying output stream as a sequence of characters.

- virtual void **writeUTF** (const std::string &value) throw (IOException, UTFDataFormatException)

  Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes.

Protected Attributes

- std::size_t **written**
- unsigned char **buffer** [8]

6.203.1 Detailed Description

A data output stream lets an application write primitive Java data types to an output stream in a portable way. An application can then use a data input stream to read the data back in.

6.203.2 Constructor & Destructor Documentation

6.203.2.1 decaf::io::DataOutputStream::DataOutputStream (OutputStream * *outputStream*, bool *own* = `false`)

Creates a new data output stream to write data to the specified underlying output stream.

Parameters:

outputStream a stream to wrap with this one.

own true if this objects owns the stream that it wraps.

6.203.2.2 virtual decaf::io::DataOutputStream::~DataOutputStream () [virtual]

6.203.3 Member Function Documentation

6.203.3.1 virtual std::size_t decaf::io::DataOutputStream::size () const [inline, virtual]

Returns the current value of the counter written, the number of bytes written to this data output stream so far. If the counter overflows, it will be wrapped to Integer.MAX_VALUE.

Returns:

the value of the written field.

6.203.3.2 virtual void decaf::io::DataOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1326).

6.203.3.3 virtual void decaf::io::DataOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream. the counter written is incremented by len.

Parameters:

buffer The array of bytes to write.

offset the position in buffer to start writing from.

len The number of bytes from the buffer to be written.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

NullPointerException if buffer is Null

Reimplemented from **decaf::io::FilterOutputStream** (p. 1326).

6.203.3.4 virtual void decaf::io::DataOutputStream::write (unsigned char *c*) throw (IOException) [virtual]

Writes a single byte to the output stream. If no exception is thrown, the counter written is incremented by 1.

Parameters:

c the byte.

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1327).

6.203.3.5 virtual void decaf::io::DataOutputStream::writeBoolean (bool *value*) throw (IOException) [virtual]

Writes a boolean to the underlying output stream as a 1-byte value. The value true is written out as the value (byte)1; the value false is written out as the value (byte)0. If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the boolean to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.6 virtual void decaf::io::DataOutputStream::writeByte (unsigned char *value*) throw (IOException) [virtual]

Writes out a byte to the underlying output stream as a 1-byte value. If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the unsigned char value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.7 virtual void decaf::io::DataOutputStream::writeBytes (const std::string & *value*) throw (IOException) [virtual]

Writes out the string to the underlying output stream as a sequence of bytes. Each character in the string is written out, in sequence, by discarding its high eight bits. If no exception is thrown, the counter written is incremented by the length of value. The value written does not include a trailing null as that is not part of the sequence of bytes, if the null is needed, then use the writeChars method.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.8 virtual void decaf::io::DataOutputStream::writeChar (char *value*) throw (IOException) [virtual]

Writes out a char to the underlying output stream as a one byte value If no exception is thrown, the counter written is incremented by 1.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.9 virtual void decaf::io::DataOutputStream::writeChars (const std::string & *value*) throw (IOException) [virtual]

Writes a string to the underlying output stream as a sequence of characters. Each character is written to the data output stream as if by the writeChar method. If no exception is thrown, the counter written is incremented by the length of value. The trailing NULL character is written by this method.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.10 virtual void decaf::io::DataOutputStream::writeDouble (double *value*) throw (IOException) [virtual]

Converts the double argument to a long using the doubleToLongBits method in class Double, and then writes that long value to the underlying output stream as an 8-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 8.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.11 virtual void decaf::io::DataOutputStream::writeFloat (float *value*) throw (IOException) [virtual]

Converts the float argument to an int using the floatToIntBits method in class Float, and then writes that int value to the underlying output stream as a 4-byte quantity, high byte first. If no exception is thrown, the counter written is incremented by 4.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.12 virtual void decaf::io::DataOutputStream::writeInt (int *value*) throw (IOException) [virtual]

Writes an int to the underlying output stream as four bytes, high byte first. If no exception is thrown, the counter written is incremented by 4.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.13 virtual void decaf::io::DataOutputStream::writeLong (long long *value*) throw (IOException) [virtual]

Writes an 64 bit long to the underlying output stream as eight bytes, high byte first. If no exception is thrown, the counter written is incremented by 8.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.14 virtual void decaf::io::DataOutputStream::writeShort (short *value*) throw (IOException) [virtual]

Writes a short to the underlying output stream as two bytes, high byte first. If no exception is thrown, the counter written is incremented by 2.

Parameters:

value the value to write.

Exceptions:

IOException *(*p. *1477)*

6.203.3.15 virtual void decaf::io::DataOutputStream::writeUnsignedShort (unsigned short *value*) throw (IOException) [virtual]

Writes a unsigned short to the bytes message stream as a 2 byte value.

Parameters:

value - unsigned short to write to the stream

Exceptions:

IOException (p. ***1477***)

6.203.3.16 virtual void decaf::io::DataOutputStream::writeUTF (const std::string & *value*) throw (IOException, UTFDataFormatException) [virtual]

Writes out the string to the underlying output stream as a modeified UTF-8 encoded sequence of bytes. The first two bytes written are indicate its encoded length followed by the rest of the string's characters encoded as modified UTF-8. The length represent the encoded length of the data not the actual length of the string.

Parameters:

value the value to write.

Exceptions:

IOException (p. ***1477***) - on a write error

UTFDataFormatException (p. ***2683***) - if encoded size if greater than 65535

6.203.4 Field Documentation

6.203.4.1 unsigned char decaf::io::DataOutputStream::buffer[8] [protected]

6.203.4.2 std::size_t decaf::io::DataOutputStream::written [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**DataOutputStream.h**

6.204 activemq::commands::DataResponse Class Reference

`#include <src/main/activemq/commands/DataResponse.h>`Inheritance diagram for activemq::commands::DataResponse:

Public Member Functions

- **DataResponse** ()
- virtual ~**DataResponse** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **DataResponse** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **DataStructure** > & **getData** () const
- virtual **Pointer**< **DataStructure** > & **getData** ()
- virtual void **setData** (const **Pointer**< **DataStructure** > &**data**)

Static Public Attributes

- static const unsigned char **ID_DATARESPONSE** = 32

Protected Member Functions

- **DataResponse** (const **DataResponse** &)
- **DataResponse** & **operator**= (const **DataResponse** &)

Protected Attributes

- **Pointer**< **DataStructure** > **data**

6.204.1 Constructor & Destructor Documentation

6.204.1.1 activemq::commands::DataResponse::DataResponse (const DataResponse &) [inline, protected]

6.204.1.2 activemq::commands::DataResponse::DataResponse ()

6.204.1.3 virtual activemq::commands::DataResponse::~DataResponse () [virtual]

6.204.2 Member Function Documentation

6.204.2.1 virtual DataResponse* activemq::commands::DataResponse::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.204.2.2 virtual void activemq::commands::DataResponse::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2232).

6.204.2.3 virtual bool activemq::commands::DataResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.204.2.4 virtual Pointer<DataStructure>& activemq::commands::DataResponse::getData () [virtual]

6.204.2.5 virtual const Pointer<DataStructure>& activemq::commands::DataResponse::getData () const [virtual]

6.204.2.6 virtual unsigned char activemq::commands::DataResponse::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.204.2.7 DataResponse& activemq::commands::DataResponse::operator= (const DataResponse &) [inline, protected]

Reimplemented from **activemq::commands::Response** (p. 2233).

6.204.2.8 virtual void activemq::commands::DataResponse::setData (const Pointer< DataStructure > & *data*) [virtual]

6.204.2.9 virtual std::string activemq::commands::DataResponse::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.204.3 Field Documentation

6.204.3.1 Pointer<DataStructure> activemq::commands::DataResponse::data [protected]

6.204.3.2 const unsigned char activemq::commands::DataResponse::ID_DATARESPONSE = 32 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataResponse.h**

6.205 activemq::wireformat::openwire::marshal::v3::DataResponseMarsh Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1135).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/DataResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual ~**DataResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.205.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1135). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.205.2 Constructor & Destructor Documentation

6.205.2.1 activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::DataResponseMar () [inline]

6.205.2.2 virtual activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::~DataResponseM () [inline, virtual]

6.205.3 Member Function Documentation

6.205.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.205.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.205.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.205.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.205.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.205.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2249).

6.205.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DataResponseMarshaller.h**

6.206 activemq::wireformat::openwire::marshal::v1::DataResponseMarsh Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1139).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/DataResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual ~**DataResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.206.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1139). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.206.2 Constructor & Destructor Documentation

6.206.2.1 activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::DataResponseMar () [inline]

6.206.2.2 virtual activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::~DataResponseM () [inline, virtual]

6.206.3 Member Function Documentation

6.206.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.206.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.206.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.206.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.206.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.206.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2254).

6.206.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2255).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DataResponseMarshaller.h**

6.207 activemq::wireformat::openwire::marshal::v2::DataResponseMarsh Class Reference

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1143).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/DataResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller:

Public Member Functions

- **DataResponseMarshaller** ()
- virtual ~**DataResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.207.1 Detailed Description

Marshaling code for Open Wire Format for **DataResponseMarshaller** (p. 1143). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.207.2 Constructor & Destructor Documentation

6.207.2.1 activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::DataResponseMar () [inline]

6.207.2.2 virtual activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::~DataResponseM () [inline, virtual]

6.207.3 Member Function Documentation

6.207.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.207.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.207.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.207.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.207.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.207.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2244).

6.207.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DataResponseMarshaller.h**

6.208 activemq::wireformat::openwire::marshal::DataStreamMarshaller Class Reference

Base class for all classes that **marshal** (p. 76) **commands** (p. 59) for Openwire.

`#include <src/main/activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::DataStreamMarshaller:

Public Member Functions

- virtual ~**DataStreamMarshaller** ()
- virtual unsigned char **getDataStructureType** () const =0

  Gets the DataStructureType that this class marshals/unmarshals.

- virtual **commands::DataStructure** * **createObject** () const =0

  Creates a new instance of the class that this class is a marshaling director for.

- virtual int **tightMarshal1** (**OpenWireFormat** *format, **commands::DataStructure** *command, **utils::BooleanStream** *bs)=0 throw (decaf::io::IOException)

  Tight Marhsal to the given stream.

- virtual void **tightMarshal2** (**OpenWireFormat** *format, **commands::DataStructure** *command, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs)=0 throw (decaf::io::IOException)

  Tight Marhsal to the given stream.

- virtual void **tightUnmarshal** (**OpenWireFormat** *format, **commands::DataStructure** *command, **decaf::io::DataInputStream** *dis, **utils::BooleanStream** *bs)=0 throw (decaf::io::IOException)

  Tight Un-marhsal to the given stream.

- virtual void **looseMarshal** (**OpenWireFormat** *format, **commands::DataStructure** *command, **decaf::io::DataOutputStream** *ds)=0 throw (decaf::io::IOException)

  Tight Marhsal to the given stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *format, **commands::DataStructure** *command, **decaf::io::DataInputStream** *dis)=0 throw (decaf::io::IOException)

  Loose Un-marhsal to the given stream.

6.208.1 Detailed Description

Base class for all classes that **marshal** (p. 76) **commands** (p. 59) for Openwire.

6.208.2 Constructor & Destructor Documentation

6.208.2.1 virtual activemq::wireformat::openwire::marshal::DataStreamMarshaller::~DataStreamMarshaller () [inline, virtual]

6.208.3 Member Function Documentation

6.208.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::DataStreamMarshaller::createObject () const [pure virtual]

Creates a new instance of the class that this class is a marshaling director for.

Returns:

newly allocated Command

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 156), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 272), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 346), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 425), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 458), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 474), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 620), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 955), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 971), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 990), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1008), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1038), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1055), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1088), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1204), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1222), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1361), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 1500), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 1521), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 1536), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 1552), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1583), **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1613), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**

(p. 1792), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 1820), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 1834), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 1849), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1907), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1936), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2008), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2095), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2119), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2136), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2186), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2207), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2223), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2252), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2289), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2305), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2357), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2497), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2603), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2718), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2737), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 160), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 187), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 276), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 291), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 322), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 350), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 396), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 429), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 446), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 462), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 478), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 604), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 624), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 959), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 975), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 994), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1012), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1042), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1059), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1079), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1096), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1113), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1144), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1208), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1226), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1288), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1369), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1446), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1496), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1513), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1528), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1544), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1563), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller

(p. 1587), **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1605), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1784), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1812), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1830), **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller** (p. 1857), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1899), **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller** (p. 1932), **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2000), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2099), **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller** (p. 2111), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2128), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2182), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2199), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2215), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242), **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller** (p. 2297), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2309), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2353), **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller** (p. 2505), **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2595), **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller** (p. 2714), **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2745), **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 152), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 268), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 283), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 314), **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 342), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 421), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 438), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 454), **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 470), **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 951), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 967), **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller** (p. 986), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1004), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1034), **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller** (p. 1051), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1071), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1092), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1200), **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** (p. 1218), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1365), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 1504), **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 1517), **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** (p. 1548),

activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1559), **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579), **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1609), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1788), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1838), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1903), **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 1928), **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2091), **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 2115), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2132), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2190), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2203), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2219), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 2293), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2313), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2361), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 2501), **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2599), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 2710), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2741).

6.208.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::DataStreamMarshaller::getDataStructureType () const [pure virtual]

Gets the DataStructureType that this class marshals/unmarshals.

Returns:

byte Id of this classes DataStructureType

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 156), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 272), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 346), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 425), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 458), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 474), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 620), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 955), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 971), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 990), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1008), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1038), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1055),

activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1075), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1088), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1109), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1140), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1204), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1222), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1284), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1361), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1450), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1500), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1521), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1536), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1552), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1567), activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 1583), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 1613), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 1792), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 1820), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 1834), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 1849), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1907), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1936), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2008), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2095), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2119), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2136), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2186), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2207), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2223), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2252), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2289), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2305), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2357), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2497), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2603), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2718), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2737), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 160), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 187), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 276), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 291), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 322), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 350), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 396), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 429), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 446), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 462), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 478), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 604), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 624), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 959), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 975), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 994), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1012),

activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1042), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1059), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1079), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1096), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1113), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1144), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1208), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1226), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1288), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1369), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1446), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1496), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1513), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1528), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1544), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1563), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1587), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1605), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 1784), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 1812), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 1830), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 1857), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 1899), activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 1932), activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2000), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2099), activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2111), activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2128), activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2182), activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 2199), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 2215), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2242), activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2297), activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2309), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2353), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2505), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 2595), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 2714), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 2745), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 152), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 179), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 268), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 283), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 314), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 342), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 388), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 421), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 438), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 454), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 470), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 596), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 616), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 951), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 967),

activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 986), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1004), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1034), **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller** (p. 1051), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1071), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1092), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1200), **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** (p. 1218), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1365), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 1504), **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 1517), **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 1532), **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** (p. 1548), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579), **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1609), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1788), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1838), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1903), **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 1928), **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2091), **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 2115), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2132), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2190), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2203), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2219), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 2293), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2313), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2361), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 2501), **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2599), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 2710), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2741).

6.208.3.3 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseMarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*) throw (decaf::io::IOException) [pure virtual]

Tight Marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

ds - DataOutputStream to **marshal** (p. 76) to

Exceptions:

IOException if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 156), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 272), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 346), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 392), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 408), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 425), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 442), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 458), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 474), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 600), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 620), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 955), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 971), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 990), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1008), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1038), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1055), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1075), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1088), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1204), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1222), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1361), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 1500), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 1521), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 1536), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 1552), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1567), **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1583), **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1613), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1792), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1820), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1834), **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** (p. 1849), **activemq::wireformat::openwire::marshal::v1::MessageMarshaller** (p. 1872), **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller** (p. 1907), **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller** (p. 1936), **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2008), **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller** (p. 2095), **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller** (p. 2119),

activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2136), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2186), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2207), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2223), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2252), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2289), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2305), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2357), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2497), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 2578), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2603), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2718), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2737), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 160), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 187), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 249), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 276), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 291), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 322), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 350), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 396), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 412), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 429), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 446), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 462), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 478), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 529), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 604), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 624), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 959), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 975), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 994), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1012), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1042), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1059), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1079), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1096), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1113), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1144), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1208), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1226), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1288), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1369), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1446), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1496), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1513), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1528), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1544), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1563), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1587), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1605), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 1784), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 1812), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller

(p. 1830), **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller** (p. 1857), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1899), **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller** (p. 1932), **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2000), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2099), **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller** (p. 2111), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2128), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2182), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2199), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2215), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242), **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller** (p. 2297), **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller** (p. 2309), **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller** (p. 2353), **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller** (p. 2505), **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2586), **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller** (p. 2595), **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller** (p. 2714), **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2745), **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 152), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller** (p. 241), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 268), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 283), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 314), **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller** (p. 342), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 388), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller** (p. 404), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller** (p. 421), **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller** (p. 438), **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 454), **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller** (p. 470), **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515), **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller** (p. 596), **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller** (p. 616), **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller** (p. 951), **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller** (p. 967), **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller** (p. 986), **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller** (p. 1004), **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller** (p. 1034), **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller** (p. 1051), **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller** (p. 1071), **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller** (p. 1092), **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller** (p. 1200), **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller** (p. 1218), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1365), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454), **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 1504), **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 1517), **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 1532), **ac-**

tivemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1548), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1559), **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579), **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1609), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1788), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1816), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1838), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 1853), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1867), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1903), **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 1928), **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2091), **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 2115), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2132), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2190), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2203), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2219), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 2293), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2313), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2361), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 2501), **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582), **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2599), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 2710), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2741).

6.208.3.4 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::looseUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException) [pure virtual]

Loose Un-marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Un-Marshal

dis - the DataInputStream to Un-Marshal from

Exceptions:

IOException if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 245), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**

(p. 347), activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller (p. 393), activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller (p. 408), activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller (p. 426), activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller (p. 443), activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller (p. 459), activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller (p. 475), activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller (p. 523), activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller (p. 601), activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller (p. 621), activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller (p. 956), activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller (p. 972), activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller (p. 991), activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller (p. 1009), activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1039), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1056), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1076), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1089), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1110), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1141), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1205), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1223), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1285), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1362), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1451), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1501), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1522), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1537), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1553), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1568), activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 1584), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 1614), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 1793), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 1821), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 1835), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 1850), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 1872), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1908), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1937), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2009), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2096), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2120), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2137), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2187), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2208), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2224), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2253), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2290), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2306), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2358), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2498), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 2578), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2604), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2719), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2738),

activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 161), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller** (p. 249), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 277), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller** (p. 351), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 397), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller** (p. 412), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller** (p. 430), **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller** (p. 447), **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 463), **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller** (p. 479), **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530), **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller** (p. 605), **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller** (p. 625), **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller** (p. 960), **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller** (p. 976), **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller** (p. 995), **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller** (p. 1013), **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller** (p. 1043), **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller** (p. 1060), **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller** (p. 1080), **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller** (p. 1097), **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1114), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1145), **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller** (p. 1209), **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller** (p. 1227), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1289), **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller** (p. 1370), **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1447), **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller** (p. 1497), **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller** (p. 1514), **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller** (p. 1529), **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller** (p. 1545), **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller** (p. 1564), **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1588), **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1606), **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller** (p. 1785), **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller** (p. 1813), **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller** (p. 1831), **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller** (p. 1858), **activemq::wireformat::openwire::marshal::v2::MessageMarshaller** (p. 1862), **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller** (p. 1900), **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller** (p. 1933), **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2001), **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller** (p. 2100), **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller** (p. 2112), **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller** (p. 2129), **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller** (p. 2183), **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller** (p. 2200), **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller** (p. 2216), **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243), **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller** (p. 2298), **ac-**

tivemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2310), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2354), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2506), activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 2586), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 2596), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 2715), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 2746), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 153), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 180), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 241), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 269), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 284), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller (p. 315), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 343), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 389), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 404), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 422), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 439), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 455), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 471), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (p. 516), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 597), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 617), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 952), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 968), activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 987), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1005), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1035), activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1052), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1072), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1093), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1106), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1137), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1201), activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1219), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1281), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1366), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1455), activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1505), activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1518), activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1533), activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1549), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1560), activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 1580), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 1610), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 1789), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 1817), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 1839), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 1854), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 1867), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 1904), activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 1929), activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2005), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2092),

activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2116), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2133), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2191), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2204), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2220), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 2294), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2314), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2362), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 2502), **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582), **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2600), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 2711), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2742).

6.208.3.5 virtual int activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal1 (OpenWireFormat * *format*, commands::DataStructure * *command*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 347), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 393), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 426), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 443), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 459), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 475), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 601), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 621), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 956), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 972), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 991), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1009),

activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller (p. 1039), activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller (p. 1056), activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller (p. 1076), activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller (p. 1089), activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller (p. 1110), activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller (p. 1141), activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller (p. 1205), activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller (p. 1223), activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller (p. 1285), activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller (p. 1362), activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller (p. 1451), activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller (p. 1501), activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1522), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1537), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1553), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1568), activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 1584), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 1614), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 1793), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 1821), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 1835), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 1850), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 1873), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1908), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1937), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2009), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2096), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2120), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2137), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2187), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2208), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2224), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2253), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2290), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2306), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2358), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2498), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 2579), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2604), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2719), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2738), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 161), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 188), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 250), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 277), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 292), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 323), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 351), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 397), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 413), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 430), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 447), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 463), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller

(p. 479), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 531), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 605), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 625), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 960), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 976), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 995), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1013), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1043), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1060), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1080), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1097), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1114), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1145), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1209), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1227), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1289), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1370), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1447), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1497), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1514), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1529), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1545), activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller (p. 1564), activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller (p. 1588), activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller (p. 1606), activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller (p. 1785), activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller (p. 1813), activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller (p. 1831), activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller (p. 1858), activemq::wireformat::openwire::marshal::v2::MessageMarshaller (p. 1863), activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller (p. 1900), activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller (p. 1933), activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller (p. 2001), activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller (p. 2100), activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller (p. 2112), activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller (p. 2129), activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller (p. 2183), activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller (p. 2200), activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller (p. 2216), activemq::wireformat::openwire::marshal::v2::ResponseMarshaller (p. 2243), activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller (p. 2298), activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller (p. 2310), activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller (p. 2354), activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller (p. 2506), activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller (p. 2587), activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller (p. 2596), activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller (p. 2715), activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller (p. 2746), activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller (p. 153), activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller (p. 180), activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller (p. 242), activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller (p. 269), activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller (p. 284), activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller

(p. 315), activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller (p. 343), activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller (p. 389), activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller (p. 405), activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller (p. 422), activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller (p. 439), activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller (p. 455), activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller (p. 471), activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller (p. 517), activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller (p. 597), activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller (p. 617), activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller (p. 952), activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller (p. 968), activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 987), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1005), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1035), activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1052), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1072), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1093), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1106), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1137), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1201), activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1219), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1281), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1366), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1455), activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1505), activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1518), activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1533), activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1549), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1560), activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 1580), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 1610), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 1789), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 1817), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 1839), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 1854), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 1868), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 1904), activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 1929), activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2005), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2092), activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2116), activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2133), activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2191), activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 2204), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 2220), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2248), activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2294), activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2314), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2362), activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2502), activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 2583), activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 2600), activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 2711), and

activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 2742).

6.208.3.6 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightMarshal2 (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Marshal

ds - the DataOutputStream to Marshal to

bs - boolean stream to **marshal** (p. 76) to.

Exceptions:

IOException if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 246), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 348), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 409), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 476), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 957), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 973), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 992), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1010), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1040), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1057), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1077), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1090), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1206), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1224), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1363), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 1502),

activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller (p. 1523), activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller (p. 1538), activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller (p. 1554), activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller (p. 1569), activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller (p. 1585), activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller (p. 1615), activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller (p. 1794), activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller (p. 1822), activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller (p. 1836), activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller (p. 1851), activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 1874), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1909), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1938), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2010), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2097), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2121), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2138), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2188), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2209), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2225), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2254), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2291), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2307), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2359), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2499), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 2579), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2605), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2720), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2739), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 162), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 189), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 250), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 278), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 293), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 324), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 352), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 398), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 413), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 431), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 448), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 464), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 480), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 532), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 606), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 626), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 961), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 977), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 996), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1014), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1044), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1061), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1081), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1098), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1115),

activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller (p. 988), activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller (p. 1006), activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller (p. 1036), activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller (p. 1053), activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller (p. 1073), activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller (p. 1094), activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller (p. 1107), activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller (p. 1138), activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller (p. 1202), activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller (p. 1220), activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1282), activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller (p. 1367), activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller (p. 1456), activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller (p. 1506), activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller (p. 1519), activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller (p. 1534), activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller (p. 1550), activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller (p. 1561), activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller (p. 1581), activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller (p. 1611), activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller (p. 1790), activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller (p. 1818), activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller (p. 1840), activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller (p. 1855), activemq::wireformat::openwire::marshal::v3::MessageMarshaller (p. 1869), activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller (p. 1905), activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller (p. 1930), activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller (p. 2006), activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller (p. 2093), activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller (p. 2117), activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller (p. 2134), activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller (p. 2192), activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller (p. 2205), activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller (p. 2221), activemq::wireformat::openwire::marshal::v3::ResponseMarshaller (p. 2249), activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller (p. 2295), activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller (p. 2315), activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller (p. 2363), activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller (p. 2503), activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller (p. 2583), activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller (p. 2601), activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller (p. 2712), and activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller (p. 2743).

6.208.3.7 virtual void activemq::wireformat::openwire::marshal::DataStreamMarshaller::tightUnmarshal (OpenWireFormat * *format*, commands::DataStructure * *command*, decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [pure virtual]

Tight Un-marhsal to the given stream.

Parameters:

format - The OpenwireFormat properties

command - the object to Un-Marshal

dis - the DataInputStream to Un-Marshal from

bs - boolean stream to unmarshal from.

Exceptions:

IOException if an error occurs.

Implemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller** (p. 247), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller** (p. 348), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller** (p. 410), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller** (p. 427), **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller** (p. 444), **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 460), **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller** (p. 476), **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526), **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller** (p. 602), **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller** (p. 622), **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller** (p. 957), **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller** (p. 973), **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller** (p. 992), **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller** (p. 1010), **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller** (p. 1040), **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller** (p. 1057), **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller** (p. 1077), **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller** (p. 1090), **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller** (p. 1206), **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller** (p. 1224), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller** (p. 1363), **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452), **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller** (p. 1502), **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller** (p. 1523), **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller** (p. 1538), **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller** (p. 1554), **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller** (p. 1569), **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1585), **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1615), **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller** (p. 1794), **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller** (p. 1822), **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller** (p. 1836), **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller** (p. 1851),

activemq::wireformat::openwire::marshal::v1::MessageMarshaller (p. 1874), activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller (p. 1909), activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller (p. 1938), activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller (p. 2010), activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller (p. 2097), activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller (p. 2121), activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller (p. 2138), activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller (p. 2188), activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller (p. 2209), activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller (p. 2225), activemq::wireformat::openwire::marshal::v1::ResponseMarshaller (p. 2255), activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller (p. 2291), activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller (p. 2307), activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller (p. 2359), activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller (p. 2499), activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller (p. 2580), activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller (p. 2605), activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller (p. 2720), activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller (p. 2739), activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller (p. 162), activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller (p. 189), activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller (p. 251), activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller (p. 278), activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller (p. 293), activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller (p. 324), activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller (p. 352), activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller (p. 398), activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller (p. 414), activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller (p. 431), activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller (p. 448), activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller (p. 464), activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller (p. 480), activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller (p. 533), activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller (p. 606), activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller (p. 626), activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller (p. 961), activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller (p. 977), activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller (p. 996), activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller (p. 1014), activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller (p. 1044), activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller (p. 1061), activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller (p. 1081), activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller (p. 1098), activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller (p. 1115), activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller (p. 1146), activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller (p. 1210), activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller (p. 1228), activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller (p. 1290), activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller (p. 1371), activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller (p. 1448), activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller (p. 1498), activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller (p. 1515), activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller (p. 1530), activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller (p. 1546),

tivemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller (p. 1282), **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller** (p. 1367), **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1456), **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller** (p. 1506), **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller** (p. 1519), **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller** (p. 1534), **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller** (p. 1550), **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller** (p. 1561), **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1581), **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1611), **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller** (p. 1790), **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller** (p. 1818), **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller** (p. 1840), **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller** (p. 1855), **activemq::wireformat::openwire::marshal::v3::MessageMarshaller** (p. 1869), **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller** (p. 1905), **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller** (p. 1930), **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2006), **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller** (p. 2093), **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller** (p. 2117), **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller** (p. 2134), **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller** (p. 2192), **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller** (p. 2205), **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller** (p. 2221), **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250), **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller** (p. 2295), **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller** (p. 2315), **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller** (p. 2363), **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller** (p. 2503), **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2584), **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller** (p. 2601), **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller** (p. 2712), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2743).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**DataStreamMarshaller.h**

6.209 activemq::commands::DataStructure Class Reference

`#include <src/main/activemq/commands/DataStructure.h>`Inheritance diagram for activemq::commands::DataStructure:

Public Member Functions

- virtual ~**DataStructure** ()
- virtual unsigned char **getDataStructureType** () const =0

  *Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.*

- virtual **DataStructure** * **cloneDataStructure** () const =0

  Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)=0

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const =0

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const =0

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

6.209.1 Constructor & Destructor Documentation

6.209.1.1 virtual activemq::commands::DataStructure::~DataStructure () [inline, virtual]

6.209.2 Member Function Documentation

6.209.2.1 virtual DataStructure* activemq::commands::DataStructure::cloneDataStructure () const [pure virtual]

Clone this obbject and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 147), **activemq::commands::ActiveMQBytesMessage** (p. 166), **activemq::commands::ActiveMQDestination** (p. 231), **activemq::commands::ActiveMQMapMessage** (p. 258), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage**

(p. 311), **activemq::commands::ActiveMQQueue** (p. 338), **activemq::commands::ActiveMQStreamMessage** (p. 377), **activemq::commands::ActiveMQTempDestination** (p. 400), **activemq::commands::ActiveMQTempQueue** (p. 416), **activemq::commands::ActiveMQTempTopic** (p. 433), **activemq::commands::ActiveMQTextMessage** (p. 450), **activemq::commands::ActiveMQTopic** (p. 466), **activemq::commands::BooleanExpression** (p. 578), **activemq::commands::BrokerError** (p. 587), **activemq::commands::BrokerId** (p. 593), **activemq::commands::BrokerInfo** (p. 609), **activemq::commands::ConnectionControl** (p. 946), **activemq::commands::ConnectionError** (p. 963), **activemq::commands::ConnectionId** (p. 982), **activemq::commands::ConnectionInfo** (p. 998), **activemq::commands::ConsumerControl** (p. 1029), **activemq::commands::ConsumerId** (p. 1046), **activemq::commands::ConsumerInfo** (p. 1064), **activemq::commands::ControlCommand** (p. 1084), **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::DestinationInfo** (p. 1195), **activemq::commands::DiscoveryEvent** (p. 1214), **activemq::commands::ExceptionResponse** (p. 1277), **activemq::commands::FlushCommand** (p. 1358), **activemq::commands::IntegerResponse** (p. 1443), **activemq::commands::JournalQueueAck** (p. 1492), **activemq::commands::JournalTopicAck** (p. 1508), **activemq::commands::JournalTrace** (p. 1525), **activemq::commands::JournalTransaction** (p. 1540), **activemq::commands::KeepAliveInfo** (p. 1556), **activemq::commands::LastPartialCommand** (p. 1576), **activemq::commands::LocalTransactionId** (p. 1601), **activemq::commands::Message** (p. 1741), **activemq::commands::MessageAck** (p. 1778), **activemq::commands::MessageDispatch** (p. 1801), **activemq::commands::MessageDispatchNotification** (p. 1824), **activemq::commands::MessageId** (p. 1844), **activemq::commands::MessagePull** (p. 1894), **activemq::commands::NetworkBridgeFilter** (p. 1925), **activemq::commands::PartialCommand** (p. 1996), **activemq::commands::ProducerAck** (p. 2087), **activemq::commands::ProducerId** (p. 2106), **activemq::commands::ProducerInfo** (p. 2123), **activemq::commands::RemoveInfo** (p. 2178), **activemq::commands::RemoveSubscriptionInfo** (p. 2194), **activemq::commands::ReplayCommand** (p. 2211), **activemq::commands::Response** (p. 2232), **activemq::commands::SessionId** (p. 2285), **activemq::commands::SessionInfo** (p. 2301), **activemq::commands::ShutdownInfo** (p. 2350), **activemq::commands::SubscriptionInfo** (p. 2492), **activemq::commands::TransactionId** (p. 2574), **activemq::commands::TransactionInfo** (p. 2590), **activemq::commands::WireFormatInfo** (p. 2702), and **activemq::commands::XATransactionId** (p. 2732).

6.209.2.2 virtual void activemq::commands::DataStructure::copyDataStructure (const DataStructure * *src*) [pure virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

6.209.2.3 virtual bool activemq::commands::DataStructure::equals (const DataStructure * *value*) const [pure virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.209.2.4 virtual unsigned char activemq::commands::DataStructure::getDataStructureType () const [pure virtual]

Get the **DataStructure** (p. 1174) Type as defined in CommandTypes.h.

Returns:

The type of the data structure

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 148), **activemq::commands::ActiveMQBytesMessage** (p. 168), **activemq::commands::ActiveMQDestination** (p. 233), **activemq::commands::ActiveMQMapMessage** (p. 260), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQStreamMessage** (p. 377), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), **activemq::commands::ActiveMQTextMessage** (p. 451), **activemq::commands::ActiveMQTopic** (p. 467), **activemq::commands::BrokerError** (p. 588), **activemq::commands::BrokerId** (p. 594), **activemq::commands::BrokerInfo** (p. 610), **activemq::commands::ConnectionControl** (p. 947), **activemq::commands::ConnectionError** (p. 964), **activemq::commands::ConnectionId** (p. 983), **activemq::commands::ConnectionInfo** (p. 999), **activemq::commands::ConsumerControl** (p. 1030), **activemq::commands::ConsumerId** (p. 1047), **activemq::commands::ConsumerInfo** (p. 1065), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::DestinationInfo** (p. 1196), **activemq::commands::DiscoveryEvent** (p. 1215), **activemq::commands::ExceptionResponse** (p. 1277), **activemq::commands::FlushCommand** (p. 1358), **activemq::commands::IntegerResponse** (p. 1443), **activemq::commands::JournalQueueAck** (p. 1492), **activemq::commands::JournalTopicAck** (p. 1509), **activemq::commands::JournalTrace** (p. 1525), **activemq::commands::JournalTransaction** (p. 1541), **activemq::commands::KeepAliveInfo** (p. 1556), **activemq::commands::LastPartialCommand** (p. 1577), **activemq::commands::LocalTransactionId** (p. 1602), **activemq::commands::Message** (p. 1743), **activemq::commands::MessageAck** (p. 1779), **activemq::commands::MessageDispatch** (p. 1802), **activemq::commands::MessageDispatchNotification** (p. 1825), **activemq::commands::MessageId** (p. 1845), **activemq::commands::MessagePull**

(p. 1895), **activemq::commands::NetworkBridgeFilter** (p. 1925), **activemq::commands::PartialCommand** (p. 1997), **activemq::commands::ProducerAck** (p. 2088), **activemq::commands::ProducerId** (p. 2107), **activemq::commands::ProducerInfo** (p. 2124), **activemq::commands::RemoveInfo** (p. 2178), **activemq::commands::RemoveSubscriptionInfo** (p. 2195), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2233), **activemq::commands::SessionId** (p. 2286), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2350), **activemq::commands::SubscriptionInfo** (p. 2493), **activemq::commands::TransactionId** (p. 2575), **activemq::commands::TransactionInfo** (p. 2591), **activemq::commands::WireFormatInfo** (p. 2703), and **activemq::commands::XATransactionId** (p. 2733).

6.209.2.5 virtual std::string activemq::commands::DataStructure::toString () const [pure virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 150), **activemq::commands::ActiveMQBytesMessage** (p. 173), **activemq::commands::ActiveMQDestination** (p. 237), **activemq::commands::ActiveMQMapMessage** (p. 266), **activemq::commands::ActiveMQMessage** (p. 281), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQQueue** (p. 340), **activemq::commands::ActiveMQStreamMessage** (p. 382), **activemq::commands::ActiveMQTempDestination** (p. 401), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), **activemq::commands::ActiveMQTextMessage** (p. 452), **activemq::commands::ActiveMQTopic** (p. 468), **activemq::commands::BaseCommand** (p. 512), **activemq::commands::BaseDataStructure** (p. 560), **activemq::commands::BooleanExpression** (p. 579), **activemq::commands::BrokerId** (p. 594), **activemq::commands::BrokerInfo** (p. 612), **activemq::commands::Command** (p. 883), **activemq::commands::ConnectionControl** (p. 948), **activemq::commands::ConnectionError** (p. 964), **activemq::commands::ConnectionId** (p. 983), **activemq::commands::ConnectionInfo** (p. 1001), **activemq::commands::ConsumerControl** (p. 1031), **activemq::commands::ConsumerId** (p. 1048), **activemq::commands::ConsumerInfo** (p. 1067), **activemq::commands::ControlCommand** (p. 1085), **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::DestinationInfo** (p. 1197), **activemq::commands::DiscoveryEvent** (p. 1215), **activemq::commands::ExceptionResponse** (p. 1278), **activemq::commands::FlushCommand** (p. 1359), **activemq::commands::IntegerResponse** (p. 1444), **activemq::commands::JournalQueueAck** (p. 1493), **activemq::commands::JournalTopicAck** (p. 1510), **activemq::commands::JournalTrace** (p. 1526), **activemq::commands::JournalTransaction** (p. 1541), **activemq::commands::KeepAliveInfo** (p. 1557), **activemq::commands::LastPartialCommand** (p. 1577), **activemq::commands::LocalTransactionId** (p. 1603), **activemq::commands::Message** (p. 1749), **activemq::commands::MessageAck**

(p. 1781), **activemq::commands::MessageDispatch** (p. 1803), **activemq::commands::MessageDispatchNotification** (p. 1827), **activemq::commands::MessageId** (p. 1846), **activemq::commands::MessagePull** (p. 1896), **activemq::commands::NetworkBridgeFilter** (p. 1926), **activemq::commands::PartialCommand** (p. 1997), **activemq::commands::ProducerAck** (p. 2088), **activemq::commands::ProducerId** (p. 2108), **activemq::commands::ProducerInfo** (p. 2125), **activemq::commands::RemoveInfo** (p. 2179), **activemq::commands::RemoveSubscriptionInfo** (p. 2196), **activemq::commands::ReplayCommand** (p. 2212), **activemq::commands::Response** (p. 2233), **activemq::commands::SessionId** (p. 2287), **activemq::commands::SessionInfo** (p. 2302), **activemq::commands::ShutdownInfo** (p. 2351), **activemq::commands::SubscriptionInfo** (p. 2494), **activemq::commands::TransactionId** (p. 2575), **activemq::commands::TransactionInfo** (p. 2592), **activemq::commands::WireFormatInfo** (p. 2708), and **activemq::commands::XATransactionId** (p. 2734).

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DataStructure.h**

6.210 decaf::util::Date Class Reference

Wrapper class around a time value in milliseconds.

`#include <src/main/decaf/util/Date.h>`

Public Member Functions

- **Date** ()
  Default constructor - sets time to now.
- **Date** (long long milliseconds)
  Constructs the date with a given time value.
- **Date** (const **Date** &source)
  Copy constructor.
- virtual ~**Date** ()
- long long **getTime** () const
  Gets the underlying time.
- void **setTime** (long long milliseconds)
  Sets the underlying time.
- bool **after** (**Date** &when) const
  Determines wether or not this date falls after the specified time.
- bool **before** (**Date** &when) const
  Determines wether or not this date falls before the specified time.
- bool **equals** (**Date** &when) const
  Determines wether or not this date is equal to the specified time.
- **Date** & **operator**= (const **Date** &source)
  Assignment operator.

Static Public Member Functions

- static long long **getCurrentTimeMilliseconds** ()
  Returns the current time in milliseconds.

6.210.1 Detailed Description

Wrapper class around a time value in milliseconds. This class is comparable to Java's java.util.Date class.

Since:

1.0

6.210.2 Constructor & Destructor Documentation

6.210.2.1 decaf::util::Date::Date ()

Default constructor - sets time to now.

6.210.2.2 decaf::util::Date::Date (long long *milliseconds*)

Constructs the date with a given time value.

Parameters:

milliseconds The time in milliseconds;

6.210.2.3 decaf::util::Date::Date (const Date & *source*)

Copy constructor.

Parameters:

source The **Date** (p. 1179) instance to copy into this one.

6.210.2.4 virtual decaf::util::Date::~Date () [virtual]

6.210.3 Member Function Documentation

6.210.3.1 bool decaf::util::Date::after (Date & *when*) const [inline]

Determines wether or not this date falls after the specified time.

Parameters:

when The date to compare

Returns:

true if this date falls after when.

6.210.3.2 bool decaf::util::Date::before (Date & *when*) const [inline]

Determines wether or not this date falls before the specified time.

Parameters:

when The date to compare

Returns:

true if this date falls before when.

6.210.3.3 bool decaf::util::Date::equals (Date & *when*) const [inline]

Determines wether or not this date is equal to the specified time.

Parameters:

when The date to compare

Returns:

true if this date is equal to when.

6.210.3.4 static long long decaf::util::Date::getCurrentTimeMilliseconds () [static]

Returns the current time in milliseconds. Comparable to Java's System.currentTimeMillis method.

Returns:

The current time in milliseconds.

Referenced by activemq::commands::Message::isExpired().

6.210.3.5 long long decaf::util::Date::getTime () const [inline]

Gets the underlying time.

Returns:

The underlying time value in milliseconds.

6.210.3.6 Date& decaf::util::Date::operator= (const Date & *source*) [inline]

Assignment operator.

6.210.3.7 void decaf::util::Date::setTime (long long *milliseconds*) [inline]

Sets the underlying time.

Parameters:

milliseconds The underlying time value in milliseconds.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Date.h**

6.211 decaf::internal::DecafRuntime Class Reference

Handles APR initialization and termination.

`#include <src/main/decaf/internal/DecafRuntime.h>`Inheritance diagram for decaf::internal::DecafRuntime:

Public Member Functions

- **DecafRuntime** ()
  Initializes the APR Runtime for a library.
- virtual ~**DecafRuntime** ()
  Terminates the APR Runtime for a library.
- apr_pool_t * **getGlobalPool** () const
  Grants access to the Global APR Pool instance that should be used when creating new threads.

6.211.1 Detailed Description

Handles APR initialization and termination.

6.211.2 Constructor & Destructor Documentation

6.211.2.1 decaf::internal::DecafRuntime::DecafRuntime ()

Initializes the APR Runtime for a library.

6.211.2.2 virtual decaf::internal::DecafRuntime::~DecafRuntime () `[virtual]`

Terminates the APR Runtime for a library.

6.211.3 Member Function Documentation

6.211.3.1 apr_pool_t* decaf::internal::DecafRuntime::getGlobalPool () const

Grants access to the Global APR Pool instance that should be used when creating new threads.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/**DecafRuntime.h**

6.212 activemq::threads::DedicatedTaskRunner Class Reference

`#include <src/main/activemq/threads/DedicatedTaskRunner.h>`Inheritance diagram for activemq::threads::DedicatedTaskRunner:

Public Member Functions

- **DedicatedTaskRunner** (**Task** *task)
- virtual ~**DedicatedTaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)

  Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()

  Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()

  *Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.*

Protected Member Functions

- virtual void **run** ()

  Run method - called by the Thread class in the context of the thread.

6.212.1 Constructor & Destructor Documentation

6.212.1.1 activemq::threads::DedicatedTaskRunner::DedicatedTaskRunner (Task * *task*)

6.212.1.2 virtual activemq::threads::DedicatedTaskRunner::~DedicatedTaskRunner () `[virtual]`

6.212.2 Member Function Documentation

6.212.2.1 virtual void activemq::threads::DedicatedTaskRunner::run () `[protected, virtual]`

Run method - called by the Thread class in the context of the thread.

Implements **decaf::lang::Runnable** (p. 2256).

6.212.2.2 virtual void activemq::threads::DedicatedTaskRunner::shutdown () [virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implements **activemq::threads::TaskRunner** (p. 2522).

6.212.2.3 virtual void activemq::threads::DedicatedTaskRunner::shutdown (unsigned int *timeout*) [virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implements **activemq::threads::TaskRunner** (p. 2522).

6.212.2.4 virtual void activemq::threads::DedicatedTaskRunner::wakeup () [virtual]

Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.

Implements **activemq::threads::TaskRunner** (p. 2523).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**DedicatedTaskRunner.h**

6.213 activemq::transport::DefaultTransportListener Class Reference

`#include <src/main/activemq/transport/DefaultTransportListener.h>`Inheritance diagram for activemq::transport::DefaultTransportListener:

Public Member Functions

- virtual ~**DefaultTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command AMQCPP_UNUSED)

  Event handler for the receipt of a command.

- virtual void **onException** (const **decaf::lang::Exception** &ex AMQCPP_UNUSED)

  *Event handler for an exception from a command **transport** (p. 67).*

- virtual void **transportInterrupted** ()

  *The **transport** (p. 67) has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()

  *The **transport** (p. 67) has resumed after an interruption.*

6.213.1 Constructor & Destructor Documentation

6.213.1.1 virtual activemq::transport::DefaultTransportListener::~DefaultTransportListener () [inline, virtual]

6.213.2 Member Function Documentation

6.213.2.1 virtual void activemq::transport::DefaultTransportListener::onCommand (const Pointer< Command > &command *AMQCPP_UNUSED*) [inline, virtual]

Event handler for the receipt of a command. The **transport** (p. 67) passes off all received **commands** (p. 59) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2608) deletes the command upon receipt.

Parameters:

command the received command object.

6.213.2.2 virtual void activemq::transport::DefaultTransportListener::onException (const decaf::lang::Exception &ex *AMQCPP_UNUSED*) [inline, virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

ex The exception.

6.213.2.3 virtual void activemq::transport::DefaultTransportListener::transportInterrupted () [inline, virtual]

The **transport** (p. 67) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2625).

6.213.2.4 virtual void activemq::transport::DefaultTransportListener::transportResumed () [inline, virtual]

The **transport** (p. 67) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2625).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**DefaultTransportListener.h**

6.214 decaf::util::concurrent::Delayed Class Reference

A mix-in style interface for marking objects that should be acted upon after a given delay.

`#include <src/main/decaf/util/concurrent/Delayed.h>`Inheritance diagram for decaf::util::concurrent::Delayed:

Public Member Functions

- virtual ~**Delayed** ()
- virtual long long **getDelay** (const **TimeUnit** &unit)=0

  Returns the remaining delay associated with this object, in the given time unit.

6.214.1 Detailed Description

A mix-in style interface for marking objects that should be acted upon after a given delay. An implementation of this interface must define a Comparable methods that provides an ordering consistent with its getDelay method.

6.214.2 Constructor & Destructor Documentation

6.214.2.1 virtual decaf::util::concurrent::Delayed::~Delayed () `[inline, virtual]`

6.214.3 Member Function Documentation

6.214.3.1 virtual long long decaf::util::concurrent::Delayed::getDelay (const TimeUnit & *unit*) `[pure virtual]`

Returns the remaining delay associated with this object, in the given time unit.

Parameters:

unit The time unit

Returns:

the remaining delay; zero or negative values indicate that the delay has already elapsed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Delayed.h**

6.215 cms::DeliveryMode Class Reference

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

`#include <src/main/cms/DeliveryMode.h>`

Public Member Functions

- virtual ~**DeliveryMode** ()

Static Public Attributes

- static const int **PERSISTENT** = 0

  *Enumeration values for **Message** (p. 1753) Delivery Mode.*

- static const int **NON_PERSISTENT** = 1

6.215.1 Detailed Description

This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages. When a client sends a `cms::Message` (p. 1753) it can mark the **Message** (p. 1753) as either Persistent or Non-Persistent. If the client feels that the **Message** (p. 1753) cannot be lost in transit it should mark it as Persistent, otherwise if it is allowable for a **Message** (p. 1753) to occasionally be lost it can mark it as Non-Persistent. This allows the Provider to balance make tradeoffs between balance and **Message** (p. 1753) throughput.

The **DeliveryMode** (p. 1188) covers only the transport of the **Message** (p. 1753) for sending client to its destination and doesn't apply to the receiving **Message** (p. 1753) consumer. The receiving Consumer can drop Message's based on configuration such as memory limits or **Message** (p. 1753) filtering.

A message is guaranteed to be delivered once and only once by a CMS provider if the delivery mode of the message is PERSISTENT and the configuration of the **Message** (p. 1753) consumer allows for it.

Since:

1.0

6.215.2 Constructor & Destructor Documentation

6.215.2.1 virtual cms::DeliveryMode::~DeliveryMode () [inline, virtual]

6.215.3 Field Documentation

6.215.3.1 const int cms::DeliveryMode::NON_PERSISTENT = 1 [static]

6.215.3.2 const int cms::DeliveryMode::PERSISTENT = 0 [static]

Enumeration values for **Message** (p. 1753) Delivery Mode.

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setCMSDeliveryMode().

The documentation for this class was generated from the following file:

- src/main/cms/**DeliveryMode.h**

6.216 cms::Destination Class Reference

A **Destination** (p. 1190) object encapsulates a provider-specific address.

`#include <src/main/cms/Destination.h>`Inheritance diagram for cms::Destination:

Public Types

- enum **DestinationType** { **TOPIC**, **QUEUE**, **TEMPORARY_TOPIC**, **TEMPORARY_QUEUE** }

Public Member Functions

- virtual ~**Destination** ()
- virtual **DestinationType** **getDestinationType** () const =0

  *Retrieve the **Destination** (p. 1190) Type for this **Destination** (p. 1190).*

- virtual **cms::Destination** * **clone** () const =0

  Creates a new instance of this destination type that is a copy of this one, and returns it.

- virtual void **copy** (const **cms::Destination** &source)=0

  *Copies the contents of the given **Destination** (p. 1190) object to this one.*

- virtual const **CMSProperties** & **getCMSProperties** () const =0

  Retrieve any properties that might be part of the destination that was specified.

6.216.1 Detailed Description

A **Destination** (p. 1190) object encapsulates a provider-specific address. There is no standard definition of a **Destination** (p. 1190) address, each provider can provide its own definition and there can be configuration data attached to the **Destination** (p. 1190) address.

All CMS **Destination** (p. 1190) objects support concurrent use.

Since:

1.0

6.216.2 Member Enumeration Documentation

6.216.2.1 enum cms::Destination::DestinationType

Enumerator:

TOPIC
QUEUE
TEMPORARY_TOPIC
TEMPORARY_QUEUE

6.216.3 Constructor & Destructor Documentation

6.216.3.1 virtual cms::Destination::~Destination () [inline, virtual]

6.216.4 Member Function Documentation

6.216.4.1 virtual cms::Destination* cms::Destination::clone () const [pure virtual]

Creates a new instance of this destination type that is a copy of this one, and returns it.

Returns:

cloned copy of this object

Implemented in **activemq::commands::ActiveMQQueue** (p. 338), **activemq::commands::ActiveMQTempQueue** (p. 416), **activemq::commands::ActiveMQTempTopic** (p. 433), and **activemq::commands::ActiveMQTopic** (p. 466).

6.216.4.2 virtual void cms::Destination::copy (const cms::Destination & *source*) [pure virtual]

Copies the contents of the given **Destination** (p. 1190) object to this one.

Parameters:

source The source **Destination** (p. 1190) object.

6.216.4.3 virtual const CMSProperties& cms::Destination::getCMSProperties () const [pure virtual]

Retrieve any properties that might be part of the destination that was specified. This is a deviation from the JMS spec but necessary due to C++ restrictions.

Returns:

A {const} reference to a **CMSProperties** (p. 853) object.

Implemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempQueue** (p. 417), **activemq::commands::ActiveMQTempTopic** (p. 434), and **activemq::commands::ActiveMQTopic** (p. 467).

6.216.4.4 virtual DestinationType cms::Destination::getDestinationType () const [pure virtual]

Retrieve the **Destination** (p. 1190) Type for this **Destination** (p. 1190).

Returns:

The **Destination** (p. 1190) Type

Implemented in **activemq::commands::ActiveMQQueue** (p. 339), **activemq::commands::ActiveMQTempQueue** (p. 418), **activemq::commands::ActiveMQTempTopic** (p. 435), and **activemq::commands::ActiveMQTopic** (p. 467).

The documentation for this class was generated from the following file:

- src/main/cms/**Destination.h**

6.217 activemq::commands::ActiveMQDestination::DestinationFilter Struct Reference

`#include <src/main/activemq/commands/ActiveMQDestination.h>`

Static Public Attributes

- static const std::string **ANY_CHILD**
- static const std::string **ANY_DESCENDENT**

6.217.1 Field Documentation

6.217.1.1 const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_CHILD [static]

6.217.1.2 const std::string activemq::commands::ActiveMQDestination::DestinationFilter::ANY_DESCENDENT [static]

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**ActiveMQDestination.h**

6.218 activemq::commands::DestinationInfo Class Reference

`#include <src/main/activemq/commands/DestinationInfo.h>`Inheritance diagram for activemq::commands::DestinationInfo:

Public Member Functions

- **DestinationInfo** ()
- virtual ~**DestinationInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **DestinationInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &**connectionId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual unsigned char **getOperationType** () const
- virtual void **setOperationType** (unsigned char **operationType**)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long **timeout**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_DESTINATIONINFO** = 8

Protected Member Functions

- **DestinationInfo** (const **DestinationInfo** &)
- **DestinationInfo** & **operator=** (const **DestinationInfo** &)

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- **Pointer< ActiveMQDestination > destination**
- unsigned char **operationType**
- long long **timeout**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**

6.218.1 Constructor & Destructor Documentation

6.218.1.1 activemq::commands::DestinationInfo::DestinationInfo (const DestinationInfo &) [inline, protected]

6.218.1.2 activemq::commands::DestinationInfo::DestinationInfo ()

6.218.1.3 virtual activemq::commands::DestinationInfo::~DestinationInfo () [virtual]

6.218.2 Member Function Documentation

6.218.2.1 virtual DestinationInfo* activemq::commands::DestinationInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.218.2.2 virtual void activemq::commands::DestinationInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.218.2.3 virtual bool activemq::commands::DestinationInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.218.2.4 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () [virtual]

6.218.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::DestinationInfo::getBrokerPath () const [virtual]

6.218.2.6 virtual Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () [virtual]

6.218.2.7 virtual const Pointer<ConnectionId>& activemq::commands::DestinationInfo::getConnectionId () const [virtual]

6.218.2.8 virtual unsigned char activemq::commands::DestinationInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.218.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () [virtual]

6.218.2.10 virtual const Pointer<ActiveMQDestination>& activemq::commands::DestinationInfo::getDestination () const [virtual]

6.218.2.11 virtual unsigned char activemq::commands::DestinationInfo::getOperationType () const [virtual]

6.218.2.12 virtual long long activemq::commands::DestinationInfo::getTimeout () const [virtual]

6.218.2.13 DestinationInfo& activemq::commands::DestinationInfo::operator= (const DestinationInfo &) [inline, protected]

6.218.2.14 virtual void activemq::commands::DestinationInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.218.2.15 virtual void activemq::commands::DestinationInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.218.2.16 virtual void activemq::commands::DestinationInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.218.2.17 virtual void activemq::commands::DestinationInfo::setOperationType (unsigned char *operationType*) [virtual]

6.218.2.18 virtual void activemq::commands::DestinationInfo::setTimeout (long long *timeout*) [virtual]

6.218.2.19 virtual std::string activemq::commands::DestinationInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.218.2.20 virtual Pointer<Command> activemq::commands::DestinationInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.218.3 Field Documentation

6.218.3.1 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::DestinationInfo::brokerPath [protected]

6.218.3.2 Pointer<ConnectionId> activemq::commands::DestinationInfo::connectionId [protected]

6.218.3.3 Pointer<ActiveMQDestination> activemq::commands::DestinationInfo::destination [protected]

6.218.3.4 const unsigned char activemq::commands::DestinationInfo::ID_DESTINATIONINFO = 8 [static]

6.218.3.5 unsigned char activemq::commands::DestinationInfo::operationType [protected]

6.218.3.6 long long activemq::commands::DestinationInfo::timeout [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DestinationInfo.h**

6.219 activemq::wireformat::openwire::marshal::v3::DestinationInfoMars Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1199).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.219.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1199). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.219.2 Constructor & Destructor Documentation

6.219.2.1 activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::DestinationInfoM () [inline]

6.219.2.2 virtual activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::~DestinationInfo () [inline, virtual]

6.219.3 Member Function Documentation

6.219.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.219.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.219.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.219.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.219.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.219.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.219.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DestinationInfoMarshaller.h**

6.220 activemq::wireformat::openwire::marshal::v1::DestinationInfoMars Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1203).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.220.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1203). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.220.2 Constructor & Destructor Documentation

6.220.2.1 activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::DestinationInfoM () [inline]

6.220.2.2 virtual activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::~DestinationInfo () [inline, virtual]

6.220.3 Member Function Documentation

6.220.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.220.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.220.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.220.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.220.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.220.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.220.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DestinationInfoMarshaller.h**

6.221 activemq::wireformat::openwire::marshal::v2::DestinationInfoMars Class Reference

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1207).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/DestinationInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller:

Public Member Functions

- **DestinationInfoMarshaller** ()
- virtual ~**DestinationInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.221.1 Detailed Description

Marshaling code for Open Wire Format for **DestinationInfoMarshaller** (p. 1207). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.221.2 Constructor & Destructor Documentation

6.221.2.1 activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::DestinationInfoM () [inline]

6.221.2.2 virtual activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::~DestinationInfo () [inline, virtual]

6.221.3 Member Function Documentation

6.221.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.221.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.221.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.221.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.221.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.221.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.221.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DestinationInfoMarshaller.h**

6.222 activemq::cmsutil::DestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

`#include <src/main/activemq/cmsutil/DestinationResolver.h>`Inheritance diagram for activemq::cmsutil::DestinationResolver:

Public Member Functions

- virtual ~**DestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** *mgr)=0

  Initializes this destination resolver for use.

- virtual void **destroy** ()=0

  Destroys any allocated resources.

- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName, bool pubSubDomain)=0 throw (cms::CMSException)

  Resolves the given name to a destination.

6.222.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.222.2 Constructor & Destructor Documentation

6.222.2.1 virtual activemq::cmsutil::DestinationResolver::~DestinationResolver () [inline, virtual]

6.222.3 Member Function Documentation

6.222.3.1 virtual void activemq::cmsutil::DestinationResolver::destroy () [pure virtual]

Destroys any allocated resources.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1262).

6.222.3.2 virtual void activemq::cmsutil::DestinationResolver::init (ResourceLifecycleManager * *mgr*) [pure virtual]

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1211)).

Parameters:

mgr the resource lifecycle manager.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1263).

6.222.3.3 virtual cms::Destination* activemq::cmsutil::DestinationResolver::resolveDestinationName (cms::Session * *session*, const std::string & *destName*, bool *pubSubDomain*) throw (cms::CMSException) [pure virtual]

Resolves the given name to a destination. If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters:

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns:

the resolved destination

Exceptions:

cms::CMSException *(*p. ***850)*** if resolution failed.

Implemented in **activemq::cmsutil::DynamicDestinationResolver** (p. 1263).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DestinationResolver.h**

6.223 activemq::commands::DiscoveryEvent Class Reference

`#include <src/main/activemq/commands/DiscoveryEvent.h>`Inheritance diagram for activemq::commands::DiscoveryEvent:

Public Member Functions

- **DiscoveryEvent** ()
- virtual ~**DiscoveryEvent** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **DiscoveryEvent** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getServiceName** () const
- virtual std::string & **getServiceName** ()
- virtual void **setServiceName** (const std::string &**serviceName**)
- virtual const std::string & **getBrokerName** () const
- virtual std::string & **getBrokerName** ()
- virtual void **setBrokerName** (const std::string &**brokerName**)

Static Public Attributes

- static const unsigned char **ID_DISCOVERYEVENT** = 40

Protected Member Functions

- **DiscoveryEvent** (const **DiscoveryEvent** &)
- **DiscoveryEvent** & **operator**= (const **DiscoveryEvent** &)

Protected Attributes

- std::string **serviceName**
- std::string **brokerName**

6.223.1 Constructor & Destructor Documentation

6.223.1.1 activemq::commands::DiscoveryEvent::DiscoveryEvent (const DiscoveryEvent &) [inline, protected]

6.223.1.2 activemq::commands::DiscoveryEvent::DiscoveryEvent ()

6.223.1.3 virtual activemq::commands::DiscoveryEvent::~DiscoveryEvent () [virtual]

6.223.2 Member Function Documentation

6.223.2.1 virtual DiscoveryEvent* activemq::commands::DiscoveryEvent::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.223.2.2 virtual void activemq::commands::DiscoveryEvent::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.223.2.3 virtual bool activemq::commands::DiscoveryEvent::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.223.2.4 virtual std::string& activemq::commands::DiscoveryEvent::getBrokerName () [virtual]

6.223.2.5 virtual const std::string& activemq::commands::DiscoveryEvent::getBrokerName () const [virtual]

6.223.2.6 virtual unsigned char activemq::commands::DiscoveryEvent::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.223.2.7 virtual std::string& activemq::commands::DiscoveryEvent::getServiceName () [virtual]

6.223.2.8 virtual const std::string& activemq::commands::DiscoveryEvent::getServiceName () const [virtual]

6.223.2.9 DiscoveryEvent& activemq::commands::DiscoveryEvent::operator= (const DiscoveryEvent &) [inline, protected]

6.223.2.10 virtual void activemq::commands::DiscoveryEvent::setBrokerName (const std::string & *brokerName*) [virtual]

6.223.2.11 virtual void activemq::commands::DiscoveryEvent::setServiceName (const std::string & *serviceName*) [virtual]

6.223.2.12 virtual std::string activemq::commands::DiscoveryEvent::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.223.3 Field Documentation

6.223.3.1 std::string activemq::commands::DiscoveryEvent::brokerName [protected]

6.223.3.2 const unsigned char activemq::commands::DiscoveryEvent::ID_DISCOVERYEVENT = 40 [static]

6.223.3.3 std::string activemq::commands::DiscoveryEvent::serviceName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**DiscoveryEvent.h**

6.224 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMars Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1217).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryEventMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.224.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1217). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.224.2 Constructor & Destructor Documentation

6.224.2.1 activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::DiscoveryEventM () [inline]

6.224.2.2 virtual activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::~DiscoveryEvent () [inline, virtual]

6.224.3 Member Function Documentation

6.224.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.224.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.224.3.3 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.224.3.4 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.224.3.5 virtual int activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.224.3.6 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.224.3.7 virtual void activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**DiscoveryEventMarshaller.h**

6.225 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMars Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1221).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryEventMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.225.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1221). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.225.2 Constructor & Destructor Documentation

6.225.2.1 activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::DiscoveryEventM () [inline]

6.225.2.2 virtual activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::~DiscoveryEvent () [inline, virtual]

6.225.3 Member Function Documentation

6.225.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.225.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.225.3.3 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.225.3.4 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.225.3.5 virtual int activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.225.3.6 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.225.3.7 virtual void activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**DiscoveryEventMarshaller.h**

6.226 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMars Class Reference

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1225).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryEventMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller:

Public Member Functions

- **DiscoveryEventMarshaller** ()
- virtual ~**DiscoveryEventMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.226.1 Detailed Description

Marshaling code for Open Wire Format for **DiscoveryEventMarshaller** (p. 1225). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.226.2 Constructor & Destructor Documentation

6.226.2.1 activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::DiscoveryEventM () [inline]

6.226.2.2 virtual activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::~DiscoveryEvent () [inline, virtual]

6.226.3 Member Function Documentation

6.226.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.226.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.226.3.3 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.226.3.4 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.226.3.5 virtual int activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.226.3.6 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.226.3.7 virtual void activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**DiscoveryEventMarshaller.h**

6.227 activemq::core::DispatchData Class Reference

Simple POCO that contains the information necessary to route a message to a specified consumer.

`#include <src/main/activemq/core/DispatchData.h>`

Public Member Functions

- **DispatchData** ()
- **DispatchData** (const **decaf::lang::Pointer< commands::ConsumerId >** &consumer, const **decaf::lang::Pointer< commands::Message >** &message)
- const **decaf::lang::Pointer< commands::ConsumerId >** & **getConsumerId** ()
- const **decaf::lang::Pointer< commands::Message >** & **getMessage** ()

6.227.1 Detailed Description

Simple POCO that contains the information necessary to route a message to a specified consumer.

6.227.2 Constructor & Destructor Documentation

6.227.2.1 activemq::core::DispatchData::DispatchData () [inline]

6.227.2.2 activemq::core::DispatchData::DispatchData (const decaf::lang::Pointer< commands::ConsumerId > & *consumer*, const decaf::lang::Pointer< commands::Message > & *message*) [inline]

6.227.3 Member Function Documentation

6.227.3.1 const decaf::lang::Pointer<commands::ConsumerId>& activemq::core::DispatchData::getConsumerId () [inline]

6.227.3.2 const decaf::lang::Pointer<commands::Message>& activemq::core::DispatchData::getMessage () [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**DispatchData.h**

6.228 activemq::core::Dispatcher Class Reference

Interface for an object responsible for dispatching messages to consumers.

`#include <src/main/activemq/core/Dispatcher.h>`Inheritance diagram for activemq::core::Dispatcher:

Public Member Functions

- virtual ~**Dispatcher** ()
- virtual void **dispatch** (const **Pointer**< **MessageDispatch** > &message)=0

  Dispatches a message to a particular consumer.

6.228.1 Detailed Description

Interface for an object responsible for dispatching messages to consumers.

6.228.2 Constructor & Destructor Documentation

6.228.2.1 virtual activemq::core::Dispatcher::~Dispatcher () `[inline, virtual]`

6.228.3 Member Function Documentation

6.228.3.1 virtual void activemq::core::Dispatcher::dispatch (const Pointer< MessageDispatch > & *message*) `[pure virtual]`

Dispatches a message to a particular consumer.

Parameters:

message - the message to be dispatched.

Implemented in **activemq::core::ActiveMQConsumer** (p. 223), and **activemq::core::ActiveMQSession** (p. 363).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Dispatcher.h**

6.229 decaf::lang::Double Class Reference

`#include <src/main/decaf/lang/Double.h>`Inheritance diagram for decaf::lang::Double:

Public Member Functions

- **Double** (double value)
- **Double** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Double** ()
- virtual int **compareTo** (const **Double** &d) const

  *Compares this **Double** (p. 1231) instance with another.*

- bool **equals** (const **Double** &d) const
- virtual bool **operator==** (const **Double** &d) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Double** &d) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const double &d) const

  *Compares this **Double** (p. 1231) instance with another.*

- bool **equals** (const double &d) const
- virtual bool **operator==** (const double &d) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const double &d) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

Answers the long value which the receiver represents.

- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (double d1, double d2)

  Compares the two specified double values.

- static long long **doubleToLongBits** (double value)

  Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout.

- static long long **doubleToRawLongBits** (double value)

  Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values.

- static bool **isInfinite** (double value)
- static bool **isNaN** (double value)
- static double **longBitsToDouble** (long long bits)

  Returns the double value corresponding to a given bit representation.

- static double **parseDouble** (const std::string value) throw (exceptions::NumberFormatException)

  Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class ***Double*** *(*p. *1231)**.*

- static std::string **toHexString** (double value)

  Returns a hexadecimal string representation of the double argument.

- static std::string **toString** (double value)

  Returns a string representation of the double argument.

- static **Double valueOf** (double value)

  Returns a ***Double*** *(*p. *1231)* *instance representing the specified double value.*

- static **Double valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  Returns a ***Double*** *(*p. *1231)* *instance that wraps a primtive double which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 64

  The size in bits of the primitive int type.

- static const double **MAX_VALUE**

  The maximum value that the primitive type can hold.

- static const double **MIN_VALUE**

  The minimum value that the primitive type can hold.

- static const double **NaN**

  *Constant for the Not a **Number** (p. 1954) Value.*

- static const double **POSITIVE_INFINITY**

  Constant for Positive Infinity.

- static const double **NEGATIVE_INFINITY**

  Constant for Negative Infinitiy.

6.229.1 Constructor & Destructor Documentation

6.229.1.1 decaf::lang::Double::Double (double *value*)

Parameters:

value - the primitve type to wrap

6.229.1.2 decaf::lang::Double::Double (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the string to convert to a primitve type to wrap

6.229.1.3 virtual decaf::lang::Double::~Double () [inline, virtual]

6.229.2 Member Function Documentation

6.229.2.1 virtual unsigned char decaf::lang::Double::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.229.2.2 static int decaf::lang::Double::compare (double *d1*, double *d2*) [static]

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: new Double(d1).compareTo(new Double(d2))

Parameters:

d1 - the first double to compare

d2 - the second double to compare

Returns:

the value 0 if d1 is numerically equal to d2; a value less than 0 if d1 is numerically less than d2; and a value greater than 0 if d1 is numerically greater than d2.

6.229.2.3 virtual int decaf::lang::Double::compareTo (const double & *d*) const [virtual]

Compares this **Double** (p. 1231) instance with another.

Parameters:

d - the **Double** (p. 1231) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< double >** (p. 899).

6.229.2.4 virtual int decaf::lang::Double::compareTo (const Double & *d*) const [virtual]

Compares this **Double** (p. 1231) instance with another.

Parameters:

d - the **Double** (p. 1231) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

6.229.2.5 static long long decaf::lang::Double::doubleToLongBits (double *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L. If the argument is negative infinity, the result is 0xfff0000000000000L. If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToLongBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters:

value - double to be converted

Returns:

the long long bits that make up the double

6.229.2.6 static long long decaf::lang::Double::doubleToRawLongBits (double *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout, preserving Not-a-Number (NaN) values. Bit 63 (the bit that is selected by the mask 0x8000000000000000LL) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000LL. If the argument is negative infinity, the result is 0xfff0000000000000LL. If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the doubleToLongBits method, doubleToRawLongBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is a long integer that, when given to the longBitsToDouble(long) method, will produce a floating-point value the same as the argument to doubleToRawLongBits.

Parameters:

value - double to be converted

Returns:

the long long bits that make up the double

6.229.2.7 virtual double decaf::lang::Double::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.229.2.8 bool decaf::lang::Double::equals (const double & *d*) const [inline, virtual]

Parameters:

d - the **Double** (p. 1231) object to compare against.

Returns:

true if the two **Double** (p. 1231) Objects have the same value.

Implements **decaf::lang::Comparable< double >** (p. 900).

6.229.2.9 bool decaf::lang::Double::equals (const Double & *d*) const [inline]

Parameters:

d - the **Double** (p. 1231) object to compare against.

Returns:

true if the two **Double** (p. 1231) Objects have the same value.

6.229.2.10 virtual float decaf::lang::Double::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.229.2.11 virtual int decaf::lang::Double::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.229.2.12 static bool decaf::lang::Double::isInfinite (double *value*) [static]

Parameters:

value - The double to check.

Returns:

true if the double is equal to infinity.

6.229.2.13 bool decaf::lang::Double::isInfinite () const

Returns:

true if the double is equal to positive infinity.

6.229.2.14 static bool decaf::lang::Double::isNaN (double *value*) [static]

Parameters:

value - The double to check.

Returns:

true if the double is equal to NaN.

6.229.2.15 bool decaf::lang::Double::isNaN () const

Returns:

true if the double is equal to NaN.

6.229.2.16 static double decaf::lang::Double::longBitsToDouble (long long *bits*) [static]

Returns the double value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double format" bit layout.

If the argument is 0x7ff0000000000000L, the result is positive infinity. If the argument is 0xfff0000000000000L, the result is negative infinity. If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xffffffffffffffffL, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Double.doubleToRawLongBits** (p. 1235) method.

Parameters:

bits - the long long bits to convert to double

Returns:

the double converted from the bits

6.229.2.17 virtual long long decaf::lang::Double::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.229.2.18 virtual bool decaf::lang::Double::operator< (const double & *d*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< double >** (p. 900).

6.229.2.19 virtual bool decaf::lang::Double::operator< (const Double & *d*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.229.2.20 virtual bool decaf::lang::Double::operator== (const double & *d*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< double >** (p. 901).

6.229.2.21 virtual bool decaf::lang::Double::operator== (const Double & *d*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

d - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.229.2.22 static double decaf::lang::Double::parseDouble (const std::string *value*) throw (exceptions::NumberFormatException) [static]

Returns a new double initialized to the value represented by the specified string, as performed by the valueOf method of class **Double** (p. 1231).

Parameters:

value - The string to parse to an double

Returns:

a double parsed from the passed string

Exceptions:

NumberFormatException

6.229.2.23 virtual short decaf::lang::Double::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.229.2.24 static std::string decaf::lang::Double::toHexString (double *value*) [static]

Returns a hexadecimal string representation of the double argument. All characters mentioned below are ASCII characters.

∗ If the argument is NaN, the result is the string "NaN". ∗ Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a double value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1440) on the exponent value. o If m is a double value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters:

value - The double to convert to a string

Returns:

the Hex formatted double string.

6.229.2.25 static std::string decaf::lang::Double::toString (double *value*) [static]

Returns a string representation of the double argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10-3 but less than 107, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m is less than 10-3 or greater than or equal to 107, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10n <= m < 10n+1; then let a be the mathematically exact quotient of m and 10n so that 1 <= a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1440).

Parameters:

value - The double to convert to a string

Returns:

the formatted double string.

6.229.2.26 std::string decaf::lang::Double::toString () const

Returns:

this **Double** (p. 1231) Object as a String Representation

6.229.2.27 static Double decaf::lang::Double::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Double** (p. 1231) instance that wraps a primtive double which is parsed from the string value passed.

Parameters:

value - the string to parse

Returns:

a new **Double** (p. 1231) instance wrapping the double parsed from value

Exceptions:

NumberFormatException on error.

6.229.2.28 static Double decaf::lang::Double::valueOf (double *value*) [static]

Returns a **Double** (p. 1231) instance representing the specified double value.

Parameters:

value - double to wrap

Returns:

new **Double** (p. 1231) instance wrapping the primitive value

6.229.3 Field Documentation

6.229.3.1 const double decaf::lang::Double::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.229.3.2 const double decaf::lang::Double::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.229.3.3 const double decaf::lang::Double::NaN [static]

Constant for the Not a **Number** (p. 1954) Value.

6.229.3.4 const double decaf::lang::Double::NEGATIVE_INFINITY [static]

Constant for Negative Infinitiy.

6.229.3.5 const double decaf::lang::Double::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.229.3.6 const int decaf::lang::Double::SIZE = 64 [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Double.h**

6.230 decaf::internal::nio::DoubleArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/DoubleArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::DoubleArrayBuffer:

Public Member Functions

- **DoubleArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **DoubleArrayBuffer** (p. 1242) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **DoubleArrayBuffer** (double *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **DoubleArrayBuffer** (p. 1242) object that wraps the given array.*

- **DoubleArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t length, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **DoubleArrayBuffer** (const **DoubleArrayBuffer** &other)

  *Create a **DoubleArrayBuffer** (p. 1242) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**DoubleArrayBuffer** ()
- virtual double * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the double array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual DoubleBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only double buffer that shares this buffer's content.

- virtual DoubleBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual DoubleBuffer * **duplicate** ()

  Creates a new double buffer that shares this buffer's content.

- virtual double **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual double **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible double array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual DoubleBuffer & **put** (double value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual DoubleBuffer & **put** (std::size_t index, double value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual DoubleBuffer * **slice** () const

  Creates a new DoubleBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.230.1 Constructor & Destructor Documentation

6.230.1.1 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **DoubleArrayBuffer** (p. 1242) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.230.1.2 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (double * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::NullPointerException)

Creates a **DoubleArrayBuffer** (p. 1242) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.230.1.3 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *length*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **DoubleArrayBuffer** (p. 1242) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

length - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.230.1.4 decaf::internal::nio::DoubleArrayBuffer::DoubleArrayBuffer (const DoubleArrayBuffer & *other*)

Create a **DoubleArrayBuffer** (p. 1242) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **DoubleArrayBuffer** (p. 1242) this one is to mirror.

6.230.1.5 virtual decaf::internal::nio::DoubleArrayBuffer::~DoubleArrayBuffer () `[virtual]`

6.230.2 Member Function Documentation

6.230.2.1 virtual double* decaf::internal::nio::DoubleArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) `[virtual]`

Returns the double array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1253).

6.230.2.2 virtual std::size_t decaf::internal::nio::DoubleArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::DoubleBuffer** (p. 1253).

6.230.2.3 virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only double buffer which the caller then owns.

Implements **decaf::nio::DoubleBuffer** (p. 1253).

6.230.2.4 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this DoubleBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1254).

6.230.2.5 virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::duplicate () [virtual]

Creates a new double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new double Buffer which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1254).

6.230.2.6 virtual double decaf::internal::nio::DoubleArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the double is to be read

Returns:

the double that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::DoubleBuffer** (p. 1256).

6.230.2.7 virtual double decaf::internal::nio::DoubleArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the double at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::DoubleBuffer** (p. 1256).

6.230.2.8 virtual bool decaf::internal::nio::DoubleArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible double array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::DoubleBuffer** (p. 1256).

6.230.2.9 virtual bool decaf::internal::nio::DoubleArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.230.2.10 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1257).

6.230.2.11 virtual DoubleBuffer& decaf::internal::nio::DoubleArrayBuffer::put (double *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::DoubleBuffer** (p. 1257).

6.230.2.12 virtual void decaf::internal::nio::DoubleArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.230.2.13 virtual DoubleBuffer* decaf::internal::nio::DoubleArrayBuffer::slice () const [virtual]

Creates a new DoubleBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create DoubleBuffer which the caller owns.

Implements **decaf::nio::DoubleBuffer** (p. 1259).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**DoubleArrayBuffer.h**

6.231 decaf::nio::DoubleBuffer Class Reference

This class defines four categories of operations upon double buffers:.

`#include <src/main/decaf/nio/DoubleBuffer.h>`Inheritance diagram for decaf::nio::DoubleBuffer:

Public Member Functions

- virtual ~**DoubleBuffer** ()
- virtual std::string **toString** () const
- virtual double * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the double array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **DoubleBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only double buffer that shares this buffer's content.

- virtual **DoubleBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **DoubleBuffer** * **duplicate** ()=0

  Creates a new double buffer that shares this buffer's content.

- virtual double **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual double **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **DoubleBuffer** & **get** (std::vector< double > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **DoubleBuffer** & **get** (double *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible double array.

- **DoubleBuffer** & **put** (**DoubleBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the doubles remaining in the given source buffer into this buffer.

- **DoubleBuffer** & **put** (const double *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers doubles into this buffer from the given source array.

- **DoubleBuffer** & **put** (std::vector< double > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source doubles array into this buffer.

- virtual **DoubleBuffer** & **put** (double value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual **DoubleBuffer** & **put** (std::size_t index, double value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual **DoubleBuffer** * **slice** () const =0

  *Creates a new **DoubleBuffer** (p. 1250) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **DoubleBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **DoubleBuffer** &value) const
- virtual bool **operator==** (const **DoubleBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **DoubleBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **DoubleBuffer** * **allocate** (std::size_t capacity)

  Allocates a new Double buffer.

- static **DoubleBuffer** * **wrap** (double *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **DoubleBuffer** (p. 1250).*

- static **DoubleBuffer** * **wrap** (std::vector< double > &buffer)

  *Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1250).*

Protected Member Functions

- **DoubleBuffer** (std::size_t capacity)

  *Creates a **DoubleBuffer** (p. 1250) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.231.1 Detailed Description

This class defines four categories of operations upon double buffers:. o Absolute and relative get and put methods that read and write single doubles; o Relative bulk get methods that transfer contiguous sequences of doubles from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of doubles from a double array or some other double buffer into this buffer o Methods for compacting, duplicating, and slicing a double buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing double array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.231.2 Constructor & Destructor Documentation

6.231.2.1 decaf::nio::DoubleBuffer::DoubleBuffer (std::size_t *capacity*) [protected]

Creates a **DoubleBuffer** (p. 1250) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 627) in doubles

6.231.2.2 virtual decaf::nio::DoubleBuffer::~DoubleBuffer () [inline, virtual]

6.231.3 Member Function Documentation

6.231.3.1 static DoubleBuffer* decaf::nio::DoubleBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in doubles

Returns:

the **DoubleBuffer** (p. 1250) that was allocated, caller owns.

6.231.3.2 virtual double* decaf::nio::DoubleBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the double array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1244).

6.231.3.3 virtual std::size_t decaf::nio::DoubleBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1245).

6.231.3.4 virtual DoubleBuffer* decaf::nio::DoubleBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only double buffer which the caller then owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1245).

6.231.3.5 virtual DoubleBuffer& decaf::nio::DoubleBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **DoubleBuffer** (p. 1250)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1246).

6.231.3.6 virtual int decaf::nio::DoubleBuffer::compareTo (const DoubleBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.231.3.7 virtual DoubleBuffer* decaf::nio::DoubleBuffer::duplicate () [pure virtual]

Creates a new double buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new double **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1246).

6.231.3.8 virtual bool decaf::nio::DoubleBuffer::equals (const DoubleBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.231.3.9 DoubleBuffer& decaf::nio::DoubleBuffer::get (double * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers doubles from this buffer into the given destination array. If there are fewer doubles remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length doubles from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length doubles remaining in this buffer

NullPointerException if the passed buffer is null.

6.231.3.10 DoubleBuffer& decaf::nio::DoubleBuffer::get (std::vector< double > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length doubles remaining in this buffer

6.231.3.11 virtual double decaf::nio::DoubleBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the double is to be read

Returns:

the double that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1246).

6.231.3.12 virtual double decaf::nio::DoubleBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the double at the current position

Exceptions:

BufferUnderflowException *(*p. ***661)*** if there no more data to return

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1247).

6.231.3.13 virtual bool decaf::nio::DoubleBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible double array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1247).

6.231.3.14 virtual bool decaf::nio::DoubleBuffer::operator< (const DoubleBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.231.3.15 virtual bool decaf::nio::DoubleBuffer::operator== (const DoubleBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.231.3.16 virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (std::size_t *index*, double *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1247).

6.231.3.17 virtual DoubleBuffer& decaf::nio::DoubleBuffer::put (double *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658****)* - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException *(*p. ***2167****)* - If this buffer is read-only

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1248).

6.231.3.18 DoubleBuffer& decaf::nio::DoubleBuffer::put (std::vector< double > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source doubles array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **DoubleBuffer** (p. 1250)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658****)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(*p. ***2167****)* - If this buffer is read-only

6.231.3.19 DoubleBuffer& decaf::nio::DoubleBuffer::put (const double * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers doubles into this buffer from the given source array. If there are more doubles to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no doubles are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which doubles are to be read

offset- The offset within the array of the first double to be read;

length - The number of doubles to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658****)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.231.3.20 DoubleBuffer& decaf::nio::DoubleBuffer::put (DoubleBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the doubles remaining in the given source buffer into this buffer. If there are more doubles remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 632), then no doubles are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() doubles from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take doubles from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer for the remaining doubles in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.231.3.21 virtual DoubleBuffer* decaf::nio::DoubleBuffer::slice () const [pure virtual]

Creates a new **DoubleBuffer** (p. 1250) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **DoubleBuffer** (p. 1250) which the caller owns.

Implemented in **decaf::internal::nio::DoubleArrayBuffer** (p. 1248).

6.231.3.22 virtual std::string decaf::nio::DoubleBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.231.3.23 static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (std::vector< double > & *buffer*) [static]

Wraps the passed STL double Vector in a **DoubleBuffer** (p. 1250). The new buffer will be backed by the given double array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **DoubleBuffer** (p. 1250) that is backed by buffer, caller owns.

6.231.3.24 static DoubleBuffer* decaf::nio::DoubleBuffer::wrap (double * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **DoubleBuffer** (p. 1250). The new buffer will be backed by the given double array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **DoubleBuffer** (p. 1250) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**DoubleBuffer.h**

6.232 decaf::lang::DYNAMIC_CAST_TOKEN Struct Reference

`#include <src/main/decaf/lang/Pointer.h>`

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.233 activemq::cmsutil::DynamicDestinationResolver Class Reference

Resolves a CMS destination name to a `Destination`.

`#include <src/main/activemq/cmsutil/DynamicDestinationResolver.h>`Inheritance diagram for activemq::cmsutil::DynamicDestinationResolver:

Data Structures

- class **SessionResolver**

  Manages maps of names to topics and queues for a single session.

Public Member Functions

- virtual ~**DynamicDestinationResolver** ()
- virtual void **init** (**ResourceLifecycleManager** *mgr)

  Initializes this destination resolver for use.

- virtual void **destroy** ()

  Destroys any allocated resources.

- virtual **cms::Destination** * **resolveDestinationName** (**cms::Session** *session, const std::string &destName, bool pubSubDomain) throw (cms::CMSException)

  Resolves the given name to a destination.

6.233.1 Detailed Description

Resolves a CMS destination name to a `Destination`.

6.233.2 Constructor & Destructor Documentation

6.233.2.1 virtual activemq::cmsutil::DynamicDestinationResolver::~DynamicDestinationResolver () [virtual]

6.233.3 Member Function Documentation

6.233.3.1 virtual void activemq::cmsutil::DynamicDestinationResolver::destroy () [virtual]

Destroys any allocated resources.

Implements **activemq::cmsutil::DestinationResolver** (p. 1211).

6.233.3.2 virtual void activemq::cmsutil::DynamicDestinationResolver::init (ResourceLifecycleManager * *mgr*) [inline, virtual]

Initializes this destination resolver for use. Ensures that any previously allocated resources are first destroyed (e.g. calls **destroy()** (p. 1262)).

Parameters:

mgr the resource lifecycle manager.

Implements **activemq::cmsutil::DestinationResolver** (p. 1211).

6.233.3.3 virtual cms::Destination* activemq::cmsutil::DynamicDestinationResolver::resolveDestinationName (cms::Session * *session*, const std::string & *destName*, bool *pubSubDomain*) throw (cms::CMSException) [virtual]

Resolves the given name to a destination. If `pubSubDomain` is true, a topic will be returned, otherwise a queue will be returned.

Parameters:

session the session for which to retrieve resolve the destination.

destName the name to be resolved.

pubSubDomain If true, the name will be resolved to a Topic, otherwise a Queue.

Returns:

the resolved destination

Exceptions:

cms::CMSException (p. ***850***) if resolution failed.

Implements **activemq::cmsutil::DestinationResolver** (p. 1212).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**DynamicDestinationResolver.h**

6.234 decaf::util::Map< K, V, COMPARATOR >::Entry Class Reference

`#include <src/main/decaf/util/Map.h>`

Public Member Functions

- **Entry** ()
- virtual ~**Entry** ()
- const K & **getKey** () const =0
- const V & **getValue** () const =0
- void **setValue** (const V &value)=0

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map< K, V, COMPARATOR >::Entry

6.234.1 Constructor & Destructor Documentation

6.234.1.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Entry::Entry () [inline]

6.234.1.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::Entry::~Entry () [inline, virtual]

6.234.2 Member Function Documentation

6.234.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> const K& decaf::util::Map< K, V, COMPARATOR >::Entry::getKey () const [pure virtual]

6.234.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> const V& decaf::util::Map< K, V, COMPARATOR >::Entry::getValue () const [pure virtual]

6.234.2.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> void decaf::util::Map< K, V, COMPARATOR >::Entry::setValue (const V & *value*) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Map.h**

6.235 decaf::io::EOFException Class Reference

`#include <src/main/decaf/io/EOFException.h>`Inheritance diagram for decaf::io::EOFException:

Public Member Functions

- **EOFException** () throw ()

  Default Constructor.

- **EOFException** (const **lang::Exception** &ex) throw ()

  Copy Constructor.

- **EOFException** (const **EOFException** &ex) throw ()

  Copy Constructor.

- **EOFException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **EOFException** (const std::exception ***cause**) throw ()

  Constructor.

- **EOFException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor.

- virtual **EOFException** * **clone** () const

  Clones this exception.

- virtual ~**EOFException** () throw ()

6.235.1 Constructor & Destructor Documentation

6.235.1.1 decaf::io::EOFException::EOFException () throw () `[inline]`

Default Constructor.

6.235.1.2 decaf::io::EOFException::EOFException (const lang::Exception & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex the exception to copy

6.235.1.3 decaf::io::EOFException::EOFException (const EOFException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.235.1.4 decaf::io::EOFException::EOFException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.235.1.5 decaf::io::EOFException::EOFException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.235.1.6 decaf::io::EOFException::EOFException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.235.1.7 virtual decaf::io::EOFException::~EOFException () throw () [inline, virtual]

6.235.2 Member Function Documentation

6.235.2.1 virtual EOFException* decaf::io::EOFException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an Exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**EOFException.h**

6.236 decaf::lang::Exception Class Reference

`#include <src/main/decaf/lang/Exception.h>`Inheritance diagram for decaf::lang::Exception:

Public Member Functions

- **Exception** () throw ()

  Default Constructor.

- **Exception** (const **Exception** &ex) throw ()

  Copy Constructor.

- **Exception** (const std::exception ∗**cause**) throw ()

  Constructor.

- **Exception** (const char ∗file, const int lineNumber, const char ∗msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **Exception** (const char ∗file, const int lineNumber, const std::exception ∗**cause**, const char ∗msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual ∼**Exception** () throw ()
- virtual std::string **getMessage** () const

  Gets the message for this exception.

- virtual const std::exception ∗ **getCause** () const

  *Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 104) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*

- virtual void **initCause** (const std::exception ∗**cause**)

  Initializes the contained cause exception with the one given.

- virtual const char ∗ **what** () const throw ()

  Implement method from std::exception.

- virtual void **setMessage** (const char ∗msg,...)

  Sets the cause for this exception.

- virtual void **setMark** (const char ∗file, const int lineNumber)

  Adds a file/line number to the stack trace.

- virtual **Exception** ∗ **clone** () const

  Clones this exception.

- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const

  Provides the stack trace for every point where this exception was caught, marked, and rethrown.

- virtual void **printStackTrace** () const

  Prints the stack trace to std::err.

- virtual void **printStackTrace** (std::ostream &stream) const

  Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const

  Gets the stack trace as one contiguous string.

- virtual **Exception** & **operator=** (const **Exception** &ex)

  Assignment operator.

Protected Member Functions

- virtual void **setStackTrace** (const std::vector< std::pair< std::string, int > > &trace)
- virtual void **buildMessage** (const char *format, va_list &vargs)

Protected Attributes

- std::string **message**

  The cause of this exception.

- std::exception * **cause**

  *The **Exception** (p. 1268) that caused this one to be thrown.*

- std::vector< std::pair< std::string, int > > **stackTrace**

  The stack trace.

6.236.1 Constructor & Destructor Documentation

6.236.1.1 decaf::lang::Exception::Exception () throw ()

Default Constructor.

Referenced by decaf::util::concurrent::BrokenBarrierException::BrokenBarrierException(), decaf::util::concurrent::CancellationException::CancellationException(), decaf::util::concurrent::ExecutionException::ExecutionException(), decaf::net::HttpRetryException::HttpRetryException(), decaf::net::MalformedURLException::MalformedURLExceptio decaf::net::ProtocolException::ProtocolException(), decaf::util::concurrent::RejectedExecutionException::RejectedEx decaf::net::SocketTimeoutException::SocketTimeoutException(), decaf::util::concurrent::TimeoutException::TimeoutException(), decaf::net::UnknownHostException::UnknownHostException(), and decaf::net::UnknownServiceException::UnknownServiceException().

6.236.1.2 decaf::lang::Exception::Exception (const Exception & *ex*) throw ()

Copy Constructor.

Parameters:

ex The `Exception` (p. 1268) instance to copy.

6.236.1.3 decaf::lang::Exception::Exception (const std::exception * *cause*) throw ()

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.236.1.4 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.236.1.5 decaf::lang::Exception::Exception (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw ()

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.236.1.6 virtual decaf::lang::Exception::~Exception () throw () [virtual]

6.236.2 Member Function Documentation

6.236.2.1 virtual void decaf::lang::Exception::buildMessage (const char * *format*, va_list & *vargs*) [protected, virtual]

Referenced by decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.236.2.2 virtual Exception* decaf::lang::Exception::clone () const [virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this **Exception** (p. 1268) object

Implements **decaf::lang::Throwable** (p. 2556).

Reimplemented in **activemq::exceptions::ActiveMQException** (p. 253), **activemq::exceptions::BrokerException** (p. 590), **decaf::io::EOFException** (p. 1267), **decaf::io::InterruptedIOException** (p. 1464), **decaf::io::IOException** (p. 1479), **decaf::io::UTFDataFormatException** (p. 2685), **decaf::lang::exceptions::ClassCastException** (p. 837), **decaf::lang::exceptions::IllegalArgumentException** (p. 1395), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1398), **decaf::lang::exceptions::IllegalStateException** (p. 1402), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1405), **decaf::lang::exceptions::InterruptedException** (p. 1461), **decaf::lang::exceptions::InvalidStateException** (p. 1476), **decaf::lang::exceptions::NoSuchElementException** (p. 1947), **decaf::lang::exceptions::NullPointerException** (p. 1953), **decaf::lang::exceptions::NumberFormatException** (p. 1959), **decaf::lang::exceptions::RuntimeException** (p. 2261), **decaf::lang::exceptions::UnsupportedOperationException** (p. 2637), **decaf::net::BindException** (p. 565), **decaf::net::ConnectException** (p. 940), **decaf::net::HttpRetryException** (p. 1392), **decaf::net::MalformedURLException** (p. 1688), **decaf::net::NoRouteToHostException** (p. 1941), **decaf::net::PortUnreachableException** (p. 2039), **decaf::net::ProtocolException** (p. 2152), **decaf::net::SocketException** (p. 2380), **decaf::net::SocketTimeoutException** (p. 2397), **decaf::net::UnknownHostException** (p. 2631), **decaf::net::UnknownServiceException** (p. 2634), **decaf::net::URISyntaxException** (p. 2667), **decaf::nio::BufferOverflowException** (p. 660), **decaf::nio::BufferUnderflowException** (p. 663), **decaf::nio::InvalidMarkException** (p. 1472), **decaf::nio::ReadOnlyBufferException** (p. 2169), **decaf::util::concurrent::BrokenBarrierException** (p. 585), **decaf::util::concurrent::CancellationException** (p. 786), **decaf::util::concurrent::ExecutionException** (p. 1293), **decaf::util::concurrent::RejectedExecutionException** (p. 2176), and **decaf::util::concurrent::TimeoutException** (p. 2561).

6.236.2.3 virtual const std::exception* decaf::lang::Exception::getCause () const [inline, virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 104) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implements **decaf::lang::Throwable** (p. 2557).

6.236.2.4 virtual std::string decaf::lang::Exception::getMessage () const [inline, virtual]

Gets the message for this exception.

Returns:

Text formatted error message

Implements **decaf::lang::Throwable** (p. 2557).

6.236.2.5 virtual std::vector< std::pair< std::string, int> > decaf::lang::Exception::getStackTrace () const [virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown. The first item in the returned vector is the first point where the mark was set (e.g. where the exception was created).

Returns:

the stack trace.

Implements **decaf::lang::Throwable** (p. 2557).

6.236.2.6 virtual std::string decaf::lang::Exception::getStackTraceString () const [virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

Implements **decaf::lang::Throwable** (p. 2557).

6.236.2.7 virtual void decaf::lang::Exception::initCause (const std::exception * *cause*) [virtual]

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

Parameters:

cause The exception that was the cause of this one.

Implements **decaf::lang::Throwable** (p. 2558).

6.236.2.8 virtual Exception& decaf::lang::Exception::operator= (const Exception & *ex*) [virtual]

Assignment operator.

Parameters:

ex const reference to another **Exception** (p. 1268)

6.236.2.9 virtual void decaf::lang::Exception::printStackTrace (std::ostream & *stream*) const [virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

Implements **decaf::lang::Throwable** (p. 2558).

6.236.2.10 virtual void decaf::lang::Exception::printStackTrace () const [virtual]

Prints the stack trace to std::err.

Implements **decaf::lang::Throwable** (p. 2558).

6.236.2.11 virtual void decaf::lang::Exception::setMark (const char * *file*, const int *lineNumber*) [virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use __FILE__).

lineNumber The line number in the calling file (use __LINE__).

Implements **decaf::lang::Throwable** (p. 2558).

Referenced by activemq::exceptions::BrokerException::BrokerException(), and decaf::lang::exceptions::NumberFormatException::NumberFormatException().

6.236.2.12 virtual void decaf::lang::Exception::setMessage (const char * *msg*, ...) [virtual]

Sets the cause for this exception.

Parameters:

msg the format string for the msg.

... params to format into the string

Referenced by activemq::exceptions::BrokerException::BrokerException().

6.236.2.13 virtual void decaf::lang::Exception::setStackTrace (const std::vector< std::pair< std::string, int > > & *trace*) [protected, virtual]

6.236.2.14 virtual const char* decaf::lang::Exception::what () const throw () [inline, virtual]

Implement method from std::exception.

Returns:

the const char* of `getMessage()` (p. 1272).

6.236.3 Field Documentation

6.236.3.1 std::exception* decaf::lang::Exception::cause [protected]

The **Exception** (p. 1268) that caused this one to be thrown.

6.236.3.2 std::string decaf::lang::Exception::message [protected]

The cause of this exception.

6.236.3.3 std::vector< std::pair< std::string, int> > decaf::lang::Exception::stackTrace [protected]

The stack trace.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Exception.h**

6.237 cms::ExceptionListener Class Reference

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1275) that is registered with the `Connection` (p. 941).

`#include <src/main/cms/ExceptionListener.h>`

Public Member Functions

- virtual ~**ExceptionListener** ()
- virtual void **onException** (const **cms::CMSException** &ex)=0

  Called when an exception occurs.

6.237.1 Detailed Description

If a CMS provider detects a serious problem, it notifies the client application through an `ExceptionListener` (p. 1275) that is registered with the `Connection` (p. 941). An exception listener allows a client to be notified of a problem asynchronously. Some connections only consume messages via the asynchronous event mechanism so they would have no other way to learn that their connection has failed.

Since:

1.0

6.237.2 Constructor & Destructor Documentation

6.237.2.1 virtual cms::ExceptionListener::~ExceptionListener () [inline, virtual]

6.237.3 Member Function Documentation

6.237.3.1 virtual void cms::ExceptionListener::onException (const cms::CMSException & *ex*) [pure virtual]

Called when an exception occurs. Once notified of an exception the caller should no longer use the resource that generated the exception.

Parameters:

ex Exception Object that occurred.

The documentation for this class was generated from the following file:

- src/main/cms/**ExceptionListener.h**

6.238 activemq::commands::ExceptionResponse Class Reference

`#include <src/main/activemq/commands/ExceptionResponse.h>`Inheritance diagram for activemq::commands::ExceptionResponse:

Public Member Functions

- **ExceptionResponse** ()
- virtual ~**ExceptionResponse** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ExceptionResponse** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **BrokerError** > & **getException** () const
- virtual **Pointer**< **BrokerError** > & **getException** ()
- virtual void **setException** (const **Pointer**< **BrokerError** > &**exception**)

Static Public Attributes

- static const unsigned char **ID_EXCEPTIONRESPONSE** = 31

Protected Member Functions

- **ExceptionResponse** (const **ExceptionResponse** &)
- **ExceptionResponse** & **operator**= (const **ExceptionResponse** &)

Protected Attributes

- **Pointer**< **BrokerError** > **exception**

6.238.1 Constructor & Destructor Documentation

6.238.1.1 activemq::commands::ExceptionResponse::ExceptionResponse (const ExceptionResponse &) [inline, protected]

6.238.1.2 activemq::commands::ExceptionResponse::ExceptionResponse ()

6.238.1.3 virtual activemq::commands::ExceptionResponse::~ExceptionResponse () [virtual]

6.238.2 Member Function Documentation

6.238.2.1 virtual ExceptionResponse* activemq::commands::ExceptionResponse::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.238.2.2 virtual void activemq::commands::ExceptionResponse::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2232).

6.238.2.3 virtual bool activemq::commands::ExceptionResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.238.2.4 virtual unsigned char activemq::commands::ExceptionResponse::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.238.2.5 virtual Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () [virtual]

6.238.2.6 virtual const Pointer<BrokerError>& activemq::commands::ExceptionResponse::getException () const [virtual]

6.238.2.7 ExceptionResponse& activemq::commands::ExceptionResponse::operator= (const ExceptionResponse &) [inline, protected]

Reimplemented from **activemq::commands::Response** (p. 2233).

6.238.2.8 virtual void activemq::commands::ExceptionResponse::setException (const Pointer< BrokerError > & *exception*) [virtual]

6.238.2.9 virtual std::string activemq::commands::ExceptionResponse::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.238.3 Field Documentation

6.238.3.1 Pointer<BrokerError> activemq::commands::ExceptionResponse::exception [protected]

6.238.3.2 const unsigned char activemq::commands::ExceptionResponse::ID_EXCEPTIONRESPONSE = 31 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ExceptionResponse.h**

6.239 activemq::wireformat::openwire::marshal::v3::ExceptionResponseM Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1279).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual ~**ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.239.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1279). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.239.2 Constructor & Destructor Documentation

6.239.2.1 activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::ExceptionRe
() [inline]

6.239.2.2 virtual activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]

6.239.3 Member Function Documentation

6.239.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.239.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::getDataStructu
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.239.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.239.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.239.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.239.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2249).

6.239.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ExceptionResponseMarshaller.h**

6.240 activemq::wireformat::openwire::marshal::v1::ExceptionResponseM Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1283).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual ~**ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.240.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1283). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.240.2 Constructor & Destructor Documentation

6.240.2.1 activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::ExceptionRe
() [inline]

6.240.2.2 virtual activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]

6.240.3 Member Function Documentation

6.240.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.240.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::getDataStructu
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.240.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.240.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.240.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.240.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2254).

6.240.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2255).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ExceptionResponseMarshaller.h**

6.241 activemq::wireformat::openwire::marshal::v2::ExceptionResponseM Class Reference

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1287).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ExceptionResponseMarshaller.h>`Inherita
diagram for activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller:

Public Member Functions

- **ExceptionResponseMarshaller** ()
- virtual ~**ExceptionResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.241.1 Detailed Description

Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1287). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.241.2 Constructor & Destructor Documentation

6.241.2.1 activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::ExceptionRe
() [inline]

6.241.2.2 virtual activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::~ExceptionR
() [inline, virtual]

6.241.3 Member Function Documentation

6.241.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::createObject
() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.241.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::getDataStructu
() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.241.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseMarshal
(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.241.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.241.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.241.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2244).

6.241.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller::tightUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ExceptionResponseMarshaller.h**

6.242 decaf::util::concurrent::ExecutionException Class Reference

`#include <src/main/decaf/util/concurrent/ExecutionException.h>`Inheritance diagram for decaf::util::concurrent::ExecutionException:

Public Member Functions

- **ExecutionException** () throw ()
  Default Constructor.
- **ExecutionException** (const **decaf::lang::Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **ExecutionException** (const **ExecutionException** &ex) throw ()
  Copy Constructor.
- **ExecutionException** (const std::exception ∗**cause**) throw ()
  Constructor.
- **ExecutionException** (const char ∗file, const int lineNumber, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **ExecutionException** (const char ∗file, const int lineNumber, const std::exception ∗**cause**, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **ExecutionException** ∗ **clone** () const
  Clones this exception.
- virtual ∼**ExecutionException** () throw ()

6.242.1 Constructor & Destructor Documentation

6.242.1.1 decaf::util::concurrent::ExecutionException::ExecutionException () throw () [inline]

Default Constructor.

6.242.1.2 decaf::util::concurrent::ExecutionException::ExecutionException (const decaf::lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex - An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.242.1.3 decaf::util::concurrent::ExecutionException::ExecutionException (const ExecutionException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.242.1.4 decaf::util::concurrent::ExecutionException::ExecutionException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.242.1.5 decaf::util::concurrent::ExecutionException::ExecutionException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - The list of primitives that are formatted into the message

6.242.1.6 decaf::util::concurrent::ExecutionException::ExecutionException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.242.1.7 virtual decaf::util::concurrent::ExecutionException::~ExecutionException () throw () [inline, virtual]

6.242.2 Member Function Documentation

6.242.2.1 virtual ExecutionException* decaf::util::concurrent::ExecutionException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an exception that is a clone of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ExecutionException.h**

6.243 decaf::util::concurrent::Executor Class Reference

An object that executes submitted **decaf.lang.Runnable** (p. 2256) tasks.

`#include <src/main/decaf/util/concurrent/Executor.h>`

Public Member Functions

- virtual ~**Executor** ()
- virtual void **execute** (Runnable *command)=0 throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException)

  Executes the given command at some time in the future.

6.243.1 Detailed Description

An object that executes submitted **decaf.lang.Runnable** (p. 2256) tasks. This interface provides a way of decoupling task submission from the mechanics of how each task will be run, including details of thread use, scheduling, etc. An `Executor` (p. 1294) is normally used instead of explicitly creating threads. For example, rather than invoking `new Thread(new(RunnableTask())).start()` for each of a set of tasks, you might use:

```
Executor (p. 1294) executor =  anExecutor;
executor->execute( new RunnableTask1() );
executor->execute( new RunnableTask2() );
...
```

However, the `Executor` (p. 1294) interface does not strictly require that execution be asynchronous. In the simplest case, an executor can run the submitted task immediately in the caller's thread:

```
class DirectExecutor : public Executor (p. 1294) {
public:

    void execute( Runnable* r ) (p. 1295) {
        r->run();
    }

}
```

More typically, tasks are executed in some thread other than the caller's thread. The executor below spawns a new thread for each task.

```
class ThreadPerTaskExecutor : public Executor (p. 1294) {
public:
    std::vector<Thread*gt; threads;

    void execute( Runnable* r ) (p. 1295) {
        threads.push_back( new Thread( r ) );
        threads.rbegin()->start();
    }
```

```
}
```

The `Executor` (p. 1294) implementations provided in this package implement **decaf.util.concurrent.ExecutorService** (p. ??), which is a more extensive interface. The **decaf.util.concurrent.ThreadPoolExecutor** (p. ??) class provides an extensible thread pool implementation. The **decaf.util.concurrentExecutor** (p. ??) class provides convenient factory methods for these Executors.

Since:

1.0

6.243.2 Constructor & Destructor Documentation

6.243.2.1 virtual decaf::util::concurrent::Executor::~Executor () [inline, virtual]

6.243.3 Member Function Documentation

6.243.3.1 virtual void decaf::util::concurrent::Executor::execute (Runnable * *command*) throw (decaf::util::concurrent::RejectedExecutionException, decaf::lang::exceptions::NullPointerException) [pure virtual]

Executes the given command at some time in the future. The command may execute in a new thread, in a pooled thread, or in the calling thread, at the discretion of the `Executor` (p. 1294) implementation.

Parameters:

command the runnable task

Exceptions:

RejectedExecutionException *(p. 2174)* if this task cannot be accepted for execution.

NullPointerException if command is null

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Executor.h**

6.244 activemq::transport::failover::FailoverTransport Class Reference

`#include <src/main/activemq/transport/failover/FailoverTransport.h>`Inheritance diagram for activemq::transport::failover::FailoverTransport:

Public Member Functions

- **FailoverTransport** ()
- virtual ~**FailoverTransport** ()
- void **reconnect** ()

  *Indicates that the **Transport** (p. 2608) needs to reconnect to another URI in its list.*

- void **add** (const std::string &uri)

  *Adds a New URI to the List of URIs this **transport** (p. 67) can Connect to.*

- virtual void **addURI** (const **List**< **URI** > &uris)

  *Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2608) is a composite of.*

- virtual void **removeURI** (const **List**< **URI** > &uris)

  *Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2608) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2608) should result in that **Transport** (p. 2608) being disposed of.*

- virtual void **start** () throw (cms::CMSException)

  *Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59).*

- virtual void **close** () throw (cms::CMSException)

  Stops the polling thread and closes the streams.

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)

  Sets the WireFormat instance to use.

- virtual void **setTransportListener** (**TransportListener** *listener)

  *Sets the observer of asynchronous events from this **transport** (p. 67).*

- virtual **TransportListener** * **getTransportListener** () const

  *Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*

- virtual bool **isFaultTolerant** () const

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- bool **isInitialized** () const

  *Returns true if the **Transport** (p. 2608) has been initialized by a BrokerInfo command.*

- void **setInitialized** (bool value)

  *Sets the initialized **state** (p. 65) of this **Transport** (p. 2608) to true.*

- virtual **Transport** * **narrow** (const std::type_info &typeId)

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.*

- virtual std::string **getRemoteAddress** () const
- virtual bool **isPending** () const
- virtual bool **iterate** ()

  *Performs the actual Reconnect operation for the **FailoverTransport** (p. 1296), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.*

- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (decaf::io::IOException)

  reconnect to another location

- long long **getTimeout** () const
- void **setTimeout** (long long value)
- long long **getInitialReconnectDelay** () const
- void **setInitialReconnectDelay** (long long value)
- long long **getMaxReconnectDelay** () const
- void **setMaxReconnectDelay** (long long value)
- long long **getBackOffMultiplier** () const
- void **setBackOffMultiplier** (long long value)
- bool **isUseExponentialBackOff** () const
- void **setUseExponentialBackOff** (bool value)
- bool **isRandomize** () const
- void **setRandomize** (bool value)
- int **getMaxReconnectAttempts** () const
- void **setMaxReconnectAttempts** (int value)

- long long **getReconnectDelay** () const
- void **setReconnectDelay** (long long value)
- bool **isBackup** () const
- void **setBackup** (bool value)
- int **getBackupPoolSize** () const
- void **setBackupPoolSize** (int value)
- bool **isTrackMessages** () const
- void **setTrackMessages** (bool value)
- int **getMaxCacheSize** () const
- void **setMaxCacheSize** (int value)

Protected Member Functions

- void **restoreTransport** (const **Pointer**< **Transport** > &transport) throw (decaf::io::IOException)

  *Given a **Transport** (p. 2608) restore the **state** (p. 65) of the Client's connection to the Broker using the data accumulated in the State Tracker.*

- void **handleTransportFailure** (const **decaf::lang::Exception** &error) throw (decaf::lang::Exception)

  *Called when this class' **TransportListener** (p. 2624) is notified of a Failure.*

Friends

- class **FailoverTransportListener**

6.244.1 Constructor & Destructor Documentation

6.244.1.1 activemq::transport::failover::FailoverTransport::FailoverTransport ()

6.244.1.2 virtual activemq::transport::failover::FailoverTransport::~FailoverTransport () [virtual]

6.244.2 Member Function Documentation

6.244.2.1 void activemq::transport::failover::FailoverTransport::add (const std::string & *uri*)

Adds a New URI to the List of URIs this **transport** (p. 67) can Connect to.

Parameters:

uri A String version of a URI to add to the URIs to **failover** (p. 69) to.

6.244.2.2 virtual void activemq::transport::failover::FailoverTransport::addURI (const List< URI > & *uris*) [virtual]

Add a URI to the list of URI's that will represent the set of Transports that this **Transport** (p. 2608) is a composite of.

Parameters:

uris The new URIs to add to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 911).

6.244.2.3 virtual void activemq::transport::failover::FailoverTransport::close () throw (cms::CMSException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

CMSException if errors occur.

Implements **cms::Closeable** (p. 838).

6.244.2.4 long long activemq::transport::failover::FailoverTransport::getBackOffMultiplier () const [inline]

6.244.2.5 int activemq::transport::failover::FailoverTransport::getBackupPoolSize () const [inline]

6.244.2.6 long long activemq::transport::failover::FailoverTransport::getInitialReconnectDelay () const [inline]

6.244.2.7 int activemq::transport::failover::FailoverTransport::getMaxCacheSize () const [inline]

6.244.2.8 int activemq::transport::failover::FailoverTransport::getMaxReconnectAttempts () const [inline]

6.244.2.9 long long activemq::transport::failover::FailoverTransport::getMaxReconnectDelay () const [inline]

6.244.2.10 long long activemq::transport::failover::FailoverTransport::getReconnectDelay () const [inline]

6.244.2.11 virtual std::string activemq::transport::failover::FailoverTransport::getRemoteAddress () const [virtual]

Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2609).

6.244.2.12 long long activemq::transport::failover::FailoverTransport::getTimeout () const [inline]

6.244.2.13 virtual TransportListener* activemq::transport::failover::FailoverTransport::getTransportListener () const [virtual]

Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Returns:

The listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2609).

6.244.2.14 void activemq::transport::failover::FailoverTransport::handleTransportFailure (const decaf::lang::Exception & *error*) throw (decaf::lang::Exception) [protected]

Called when this class' **TransportListener** (p. 2624) is notified of a Failure.

Parameters:

error - The CMS Exception that was thrown.

Exceptions:

Exception if an error occurs.

6.244.2.15 bool activemq::transport::failover::FailoverTransport::isBackup () const [inline]

6.244.2.16 virtual bool activemq::transport::failover::FailoverTransport::isClosed () const [inline, virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Implements **activemq::transport::Transport** (p. 2609).

6.244.2.17 virtual bool activemq::transport::failover::FailoverTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2610).

6.244.2.18 virtual bool activemq::transport::failover::FailoverTransport::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2610).

6.244.2.19 bool activemq::transport::failover::FailoverTransport::isInitialized () const [inline]

Returns true if the **Transport** (p. 2608) has been initialized by a BrokerInfo command.

Returns:

true if the **Transport** (p. 2608) has been initialized by a BrokerInfo command.

6.244.2.20 virtual bool activemq::transport::failover::FailoverTransport::isPending () const [virtual]

Returns:

true if there is a need for the iterate method to be called by this classes task runner.

Implements **activemq::threads::CompositeTask** (p. 906).

6.244.2.21 bool activemq::transport::failover::FailoverTransport::isRandomize () const [inline]

6.244.2.22 bool activemq::transport::failover::FailoverTransport::isTrackMessages () const [inline]

6.244.2.23 bool activemq::transport::failover::FailoverTransport::isUseExponentialBackOff () const [inline]

6.244.2.24 virtual bool activemq::transport::failover::FailoverTransport::iterate () [virtual]

Performs the actual Reconnect operation for the **FailoverTransport** (p. 1296), when a connection is made this method returns false to indicate it doesn't need to run again, otherwise it returns true to indicate its still trying to connect.

Returns:

false to indicate a connection, true to indicate it needs to try again.

Implements **activemq::threads::Task** (p. 2520).

6.244.2.25 virtual Transport* activemq::transport::failover::FailoverTransport::narrow (const std::type_info & *typeId*) [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2610).

References activemq::transport::Transport::narrow().

6.244.2.26 virtual void activemq::transport::failover::FailoverTransport::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2610).

6.244.2.27 virtual void activemq::transport::failover::FailoverTransport::reconnect (const decaf::net::URI & *uri*) throw (decaf::io::IOException) [virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implements **activemq::transport::Transport** (p. 2611).

6.244.2.28 void activemq::transport::failover::FailoverTransport::reconnect ()

Indicates that the **Transport** (p. 2608) needs to reconnect to another URI in its list.

6.244.2.29 virtual void activemq::transport::failover::FailoverTransport::removeURI (const List< URI > & *uris*) [virtual]

Remove a URI from the set of URI's that represents the set of Transports that this **Transport** (p. 2608) is composed of, removing a URI for which the composite has created a connected **Transport** (p. 2608) should result in that **Transport** (p. 2608) being disposed of.

Parameters:

uris The new URIs to remove to the set this composite maintains.

Implements **activemq::transport::CompositeTransport** (p. 912).

6.244.2.30 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.

timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2611).

6.244.2.31 virtual Pointer<Response> activemq::transport::failover::FailoverTransport::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2612).

6.244.2.32 void activemq::transport::failover::FailoverTransport::restoreTransport (const Pointer< Transport > & *transport*) throw (decaf::io::IOException) [protected]

Given a **Transport** (p. 2608) restore the **state** (p. 65) of the Client's connection to the Broker using the data accumulated in the State Tracker.

Parameters:

transport *(p. 67)* The new **Transport** (p. 2608) connected to the Broker.

Exceptions:

IOException if an errors occurs while restoring the old **state** (p. 65).

6.244.2.33 void activemq::transport::failover::FailoverTransport::setBackOffMultiplier (long long *value*) [inline]

6.244.2.34 void activemq::transport::failover::FailoverTransport::setBackup (bool *value*) [inline]

6.244.2.35 void activemq::transport::failover::FailoverTransport::setBackupPoolSize (int *value*) [inline]

6.244.2.36 void activemq::transport::failover::FailoverTransport::setInitialized (bool *value*) [inline]

Sets the initialized **state** (p. 65) of this **Transport** (p. 2608) to true.

Parameters:

value - true if this **Transport** (p. 2608) has been initialized.

6.244.2.37 void activemq::transport::failover::FailoverTransport::setInitialReconnectDelay (long long *value*) `[inline]`

6.244.2.38 void activemq::transport::failover::FailoverTransport::setMaxCacheSize (int *value*) `[inline]`

6.244.2.39 void activemq::transport::failover::FailoverTransport::setMaxReconnectAttempts (int *value*) `[inline]`

6.244.2.40 void activemq::transport::failover::FailoverTransport::setMaxReconnectDelay (long long *value*) `[inline]`

6.244.2.41 void activemq::transport::failover::FailoverTransport::setRandomize (bool *value*) `[inline]`

6.244.2.42 void activemq::transport::failover::FailoverTransport::setReconnectDelay (long long *value*) `[inline]`

6.244.2.43 void activemq::transport::failover::FailoverTransport::setTimeout (long long *value*) `[inline]`

6.244.2.44 void activemq::transport::failover::FailoverTransport::setTrackMessages (bool *value*) `[inline]`

6.244.2.45 virtual void activemq::transport::failover::FailoverTransport::setTransportListener (TransportListener * *listener*) `[virtual]`

Sets the observer of asynchronous events from this **transport** (p. 67).

Parameters:

listener the listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2612).

6.244.2.46 void activemq::transport::failover::FailoverTransport::setUseExponentialBackOff (bool *value*) `[inline]`

6.244.2.47 virtual void activemq::transport::failover::FailoverTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*) `[inline, virtual]`

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode **commands** (p. 59).

6.244.2.48 virtual void activemq::transport::failover::FailoverTransport::start () throw (cms::CMSException) [virtual]

Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this **transport** (p. 67) has already been closed.

Implements **cms::Startable** (p. 2413).

6.244.3 Friends And Related Function Documentation

6.244.3.1 friend class FailoverTransportListener [friend]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransport.h**

6.245 activemq::transport::failover::FailoverTransportFactory Class Reference

Creates an instance of a **FailoverTransport** (p. 1296).

`#include <src/main/activemq/transport/failover/FailoverTransportFactory.h>`Inheritance diagram for activemq::transport::failover::FailoverTransportFactory:

Public Member Functions

- virtual ~**FailoverTransportFactory** ()
- virtual **Pointer**< **Transport** > **create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.*

- virtual **Pointer**< **Transport** > **createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (const **decaf::net::URI** &location, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

6.245.1 Detailed Description

Creates an instance of a **FailoverTransport** (p. 1296).

Since:

3.0

6.245.2 Constructor & Destructor Documentation

6.245.2.1 virtual activemq::transport::failover::FailoverTransportFactory::~FailoverTransportFactory () [inline, virtual]

6.245.3 Member Function Documentation

6.245.3.1 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::create (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2614).

6.245.3.2 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::createComposite (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2615).

6.245.3.3 virtual Pointer<Transport> activemq::transport::failover::FailoverTransportFactory::doCreateComposite (const decaf::net::URI & *location*, const decaf::util::Properties & *properties*) throw (exceptions::ActiveMQException) [protected, virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to.

properties - Properties to apply to the **transport** (p. 67).

Returns:

Pointer to a new **FailoverTransport** (p. 1296) instance.

Exceptions:

ActiveMQexception if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportFactory.h**

6.246 activemq::transport::failover::FailoverTransportListener Class Reference

Utility class used by the **Transport** (p. 2608) to perform the work of responding to events from the active **Transport** (p. 2608).

`#include <src/main/activemq/transport/failover/FailoverTransportListener.h>`Inheritance diagram for activemq::transport::failover::FailoverTransportListener:

Public Member Functions

- **FailoverTransportListener** (**FailoverTransport** *parent)
- virtual ~**FailoverTransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

  Event handler for the receipt of a command.

- virtual void **onException** (const **decaf::lang::Exception** &ex)

  Event handler for an exception from a command ***transport*** *(p. 67).*

- virtual void **transportInterrupted** ()

  The ***transport*** *(p. 67) has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()

  The ***transport*** *(p. 67) has resumed after an interruption.*

6.246.1 Detailed Description

Utility class used by the **Transport** (p. 2608) to perform the work of responding to events from the active **Transport** (p. 2608).

Since:

3.0

6.246.2 Constructor & Destructor Documentation

6.246.2.1 activemq::transport::failover::FailoverTransportListener::FailoverTransportListener (FailoverTransport * *parent*)

6.246.2.2 virtual activemq::transport::failover::FailoverTransportListener::~FailoverTransportListener () [virtual]

6.246.3 Member Function Documentation

6.246.3.1 virtual void activemq::transport::failover::FailoverTransportListener::onCommand (const Pointer< Command > & *command*) [virtual]

Event handler for the receipt of a command. The **transport** (p. 67) passes off all received **commands** (p. 59) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2608) deletes the command upon receipt.

Parameters:

command the received command object.

Implements **activemq::transport::TransportListener** (p. 2624).

6.246.3.2 virtual void activemq::transport::failover::FailoverTransportListener::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

ex The exception.

Implements **activemq::transport::TransportListener** (p. 2625).

6.246.3.3 virtual void activemq::transport::failover::FailoverTransportListener::transportInterrupted () [virtual]

The **transport** (p. 67) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2625).

6.246.3.4 virtual void activemq::transport::failover::FailoverTransportListener::transportResumed () [virtual]

The **transport** (p. 67) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2625).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**FailoverTransportListener.h**

6.247 decaf::util::logging::Filter Class Reference

A **Filter** (p. 1314) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.

`#include <src/main/decaf/util/logging/Filter.h>`

Public Member Functions

- virtual ~**Filter** ()
- virtual bool **isLoggable** (const **LogRecord** &record) const =0

  Check if a given log record should be published.

6.247.1 Detailed Description

A **Filter** (p. 1314) can be used to provide fine grain control over what is logged, beyond the control provided by log levels. Each **Logger** (p. 1623) and each **Handler** (p. 1382) can have a filter associated with it. The **Logger** (p. 1623) or **Handler** (p. 1382) will call the isLoggable method to check if a given **LogRecord** (p. 1645) should be published. If isLoggable returns false, the **LogRecord** (p. 1645) will be discarded.

6.247.2 Constructor & Destructor Documentation

6.247.2.1 virtual decaf::util::logging::Filter::~Filter () [inline, virtual]

6.247.3 Member Function Documentation

6.247.3.1 virtual bool decaf::util::logging::Filter::isLoggable (const LogRecord & *record*) const [pure virtual]

Check if a given log record should be published.

Parameters:

record the `LogRecord` (p. 1645) to check.

Returns:

true if the record is loggable.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Filter.h**

6.248 decaf::io::FilterInputStream Class Reference

A **FilterInputStream** (p. 1315) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.

`#include <src/main/decaf/io/FilterInputStream.h>`Inheritance diagram for decaf::io::FilterInputStream:

Public Member Functions

- **FilterInputStream** (**InputStream** *__inputStream__, bool **own**=false)

  *Constructor to create a wrapped **InputStream** (p. 1406).*

- virtual ~**FilterInputStream** ()
- virtual std::size_t **available** () const throw (IOException)

  Returns the number of bytes that can be read from this input stream without blocking.

- virtual unsigned char **read** () throw (IOException)

  Reads the next byte of data from this input stream.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (IOException, lang::exceptions::NullPointerException)

  Reads up to len bytes of data from this input stream into an array of bytes.

- virtual void **close** () throw (lang::Exception)

  *Close the Stream, the **FilterOutputStream** (p. 1322) simply calls the close method of the underlying stream.*

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

  Determines if this input stream supports the mark and reset methods.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **InputStream** * **inputStream**
- **util::concurrent::Mutex mutex**
- bool **own**
- volatile bool **closed**

6.248.1 Detailed Description

A **FilterInputStream** (p. 1315) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality. The class **FilterInputStream** (p. 1315) itself simply overrides all methods of **InputStream** (p. 1406) with versions that pass all requests to the contained input stream. Subclasses of **FilterInputStream** (p. 1315) may further override some of these methods and may also provide additional methods and fields.

6.248.2 Constructor & Destructor Documentation

6.248.2.1 decaf::io::FilterInputStream::FilterInputStream (InputStream * *inputStream*, bool *own* = `false`) `[inline]`

Constructor to create a wrapped **InputStream** (p. 1406).

Parameters:

inputStream the stream to wrap and filter

own indicates if we own the stream object, defaults to false

6.248.2.2 virtual decaf::io::FilterInputStream::~FilterInputStream () `[inline, virtual]`

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.248.3 Member Function Documentation

6.248.3.1 virtual std::size_t decaf::io::FilterInputStream::available () const throw (IOException) [inline, virtual]

Returns the number of bytes that can be read from this input stream without blocking. This method simply performs in.available() and returns the result.

Returns:

the number of bytes available without blocking.

Implements **decaf::io::InputStream** (p. 1407).

Reimplemented in **decaf::io::BufferedInputStream** (p. 634).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.2 virtual void decaf::io::FilterInputStream::close () throw (lang::Exception) [inline, virtual]

Close the Stream, the **FilterOutputStream** (p. 1322) simply calls the close method of the underlying stream.

Exceptions:

Exception

Implements **decaf::io::Closeable** (p. 840).

Reimplemented in **decaf::io::BufferedInputStream** (p. 635).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.3 virtual bool decaf::io::FilterInputStream::isClosed () const [inline, protected, virtual]

Returns:

true if this stream has been closed.

6.248.3.4 virtual void decaf::io::FilterInputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.248.3.5 virtual void decaf::io::FilterInputStream::mark (int readLimit) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit The max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1407).

Reimplemented in **decaf::io::BufferedInputStream** (p. 635).

References DECAF_CATCHALL_NOTHROW.

6.248.3.6 virtual bool decaf::io::FilterInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1407).

Reimplemented in **decaf::io::BufferedInputStream** (p. 635).

References DECAF_CATCHALL_NOTHROW.

6.248.3.7 virtual void decaf::io::FilterInputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.248.3.8 virtual void decaf::io::FilterInputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.248.3.9 virtual int decaf::io::FilterInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [inline, virtual]

Reads up to len bytes of data from this input stream into an array of bytes. This method blocks until some input is available. This method simply performs in.read(b, len) and returns the result.

Parameters:

buffer (out) the target buffer.

offset the position to start reading in the passed buffer.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException *(*p. ***1477****)* thrown if an error occurs.

NullPointerException

Implements **decaf::io::InputStream** (p. 1408).

Reimplemented in **activemq::io::LoggingInputStream** (p. 1633), **decaf::io::BufferedInputStream** (p. 635), and **decaf::io::DataInputStream** (p. 1118).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.10 virtual unsigned char decaf::io::FilterInputStream::read () throw (IOException) [inline, virtual]

Reads the next byte of data from this input stream. The value byte is returned as an unsigned char in the range 0 to 255. If no byte is available because the end of the stream has been reached, the value -1 is returned. This method blocks until input data is available, the end of the stream is detected, or an exception is thrown. This method simply performs in.read() and returns the result.

Returns:

The next byte.

Exceptions:

IOException *(*p. ***1477****)* thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

Reimplemented in **activemq::io::LoggingInputStream** (p. 1634), and **decaf::io::BufferedInputStream** (p. 636).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.11 virtual void decaf::io::FilterInputStream::reset () throw (IOException) [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1477) might be thrown. * If such an **IOException** (p. 1477) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1477). * If an **IOException** (p. 1477) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException *(p. 1477)*

Implements **decaf::io::InputStream** (p. 1408).

Reimplemented in **decaf::io::BufferedInputStream** (p. 636).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.12 virtual std::size_t decaf::io::FilterInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [inline, virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1406) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total butes skipped

Exceptions:

IOException *(p. 1477)* if an error occurs

Implements **decaf::io::InputStream** (p. 1409).

Reimplemented in **decaf::io::BufferedInputStream** (p. 636), and **decaf::io::DataInputStream** (p. 1124).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.248.3.13 virtual void decaf::io::FilterInputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.248.3.14 virtual void decaf::io::FilterInputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.248.3.15 virtual void decaf::io::FilterInputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.248.4 Field Documentation

6.248.4.1 volatile bool decaf::io::FilterInputStream::closed [protected]

6.248.4.2 InputStream* decaf::io::FilterInputStream::inputStream [protected]

6.248.4.3 util::concurrent::Mutex decaf::io::FilterInputStream::mutex [protected]

6.248.4.4 bool decaf::io::FilterInputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterInputStream.h**

6.249 decaf::io::FilterOutputStream Class Reference

This class is the superclass of all classes that filter output streams.

`#include <src/main/decaf/io/FilterOutputStream.h>`Inheritance diagram for decaf::io::FilterOutputStream:

Public Member Functions

- **FilterOutputStream** (**OutputStream** ∗**outputStream**, bool **own**=false)

  Constructor, creates a wrapped output stream.

- virtual ∼**FilterOutputStream** ()
- virtual void **write** (unsigned char c) throw (IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char ∗buffer, std::size_t offset DECAF_UNUSED, std::size_t len) throw (IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (IOException)

  Flushes any pending writes in this output stream.

- virtual void **close** () throw (lang::Exception)

  Close the Stream.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

Protected Member Functions

- virtual bool **isClosed** () const

Protected Attributes

- **OutputStream** * **outputStream**
- **util::concurrent::Mutex mutex**
- bool **own**
- volatile bool **closed**

6.249.1 Detailed Description

This class is the superclass of all classes that filter output streams. These streams sit on top of an already existing output stream (the underlying output stream) which it uses as its basic sink of data, but possibly transforming the data along the way or providing additional functionality.

The class **FilterOutputStream** (p. 1322) itself simply overrides all methods of **OutputStream** (p. 1992) with versions that pass all requests to the underlying output stream. Subclasses of **FilterOutputStream** (p. 1322) may further override some of these methods as well as provide additional methods and fields.

Due to the lack of garbage collection in C++ a design decision was made to add a boolean parameter to the constructor indicating if the wrapped `InputStream (p. 1406)` is owned by this object. That way creation of the underlying stream can occur in a Java like way. Ex:

DataOutputStream (p. 1125) os = new **DataOutputStream** (p. 1125)(new OutputStream(), true)

6.249.2 Constructor & Destructor Documentation

6.249.2.1 decaf::io::FilterOutputStream::FilterOutputStream (OutputStream * *outputStream*, bool *own* = `false`) [inline]

Constructor, creates a wrapped output stream.

Parameters:

outputStream the **OutputStream** (p. 1992) to wrap

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.249.2.2 virtual decaf::io::FilterOutputStream::~FilterOutputStream () [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.249.3 Member Function Documentation

6.249.3.1 virtual void decaf::io::FilterOutputStream::close () throw (lang::Exception) [inline, virtual]

Close the Stream. The close method of **FilterOutputStream** (p. 1322) calls its flush method, and then calls the close method of its underlying output stream, it then destroys the output stream if it is the owner.

Exceptions:

Exception

Implements **decaf::io::Closeable** (p. 840).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 639).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.249.3.2 virtual void decaf::io::FilterOutputStream::flush () throw (IOException) [inline, virtual]

Flushes any pending writes in this output stream. The flush method of **FilterOutputStream** (p. 1322) calls the flush method of its underlying output stream

Exceptions:

IOException *(p. 1477)*

Implements **decaf::io::OutputStream** (p. 1992).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 639).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.249.3.3 virtual bool decaf::io::FilterOutputStream::isClosed () const [inline, protected, virtual]

Returns:

true if this stream has been closed.

6.249.3.4 virtual void decaf::io::FilterOutputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.249.3.5 virtual void decaf::io::FilterOutputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.249.3.6 virtual void decaf::io::FilterOutputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.249.3.7 virtual void decaf::io::FilterOutputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.249.3.8 virtual void decaf::io::FilterOutputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.249.3.9 virtual void decaf::io::FilterOutputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.249.3.10 virtual void decaf::io::FilterOutputStream::write (const unsigned char * *buffer*, std::size_t offset *DECAF_UNUSED*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [inline, virtual]

Writes an array of bytes to the output stream. The write method of **FilterOutputStream** (p. 1322) calls the write method of one argument on each byte to output.

Parameters:

buffer The array of bytes to write.

offset,the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 1993).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 1635), **decaf::io::BufferedOutputStream** (p. 639), and **decaf::io::DataOutputStream** (p. 1127).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.249.3.11 virtual void decaf::io::FilterOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [inline, virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

Reimplemented in **decaf::io::BufferedOutputStream** (p. 640), and **decaf::io::DataOutputStream** (p. 1127).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.249.3.12 virtual void decaf::io::FilterOutputStream::write (unsigned char *c*) throw (IOException) [inline, virtual]

Writes a single byte to the output stream. The write method of **FilterOutputStream** (p. 1322) calls the write method of its underlying output stream, that is, it performs out.write(b).

Parameters:

c the byte.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

Reimplemented in **activemq::io::LoggingOutputStream** (p. 1636), **decaf::io::BufferedOutputStream** (p. 640), and **decaf::io::DataOutputStream** (p. 1127).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.249.4 Field Documentation

6.249.4.1 volatile bool decaf::io::FilterOutputStream::closed [protected]

6.249.4.2 util::concurrent::Mutex decaf::io::FilterOutputStream::mutex [protected]

6.249.4.3 OutputStream* decaf::io::FilterOutputStream::outputStream [protected]

6.249.4.4 bool decaf::io::FilterOutputStream::own [protected]

The documentation for this class was generated from the following file:

- src/main/decaf/io/**FilterOutputStream.h**

6.250 decaf::lang::Float Class Reference

`#include <src/main/decaf/lang/Float.h>`Inheritance diagram for decaf::lang::Float:

Public Member Functions

- **Float** (float value)
- **Float** (double value)
- **Float** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Float** ()
- virtual int **compareTo** (const **Float** &f) const

  *Compares this **Float** (p. 1328) instance with another.*

- bool **equals** (const **Float** &f) const
- virtual bool **operator==** (const **Float** &f) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Float** &f) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const float &f) const

  *Compares this **Float** (p. 1328) instance with another.*

- bool **equals** (const float &f) const
- virtual bool **operator==** (const float &f) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const float &f) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

- bool **isInfinite** () const
- bool **isNaN** () const

Static Public Member Functions

- static int **compare** (float f1, float f2)

  Compares the two specified double values.

- static int **floatToIntBits** (float value)

  Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout.

- static int **floatToRawIntBits** (float value)

  Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values.

- static float **intBitsToFloat** (int bits)

  Returns the float value corresponding to a given bit representation.

- static bool **isInfinite** (float value)
- static bool **isNaN** (float value)
- static float **parseFloat** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1328).*

- static std::string **toHexString** (float value)

  Returns a hexadecimal string representation of the float argument.

- static std::string **toString** (float value)

  Returns a string representation of the float argument.

- static **Float valueOf** (float value)

  *Returns a **Float** (p. 1328) instance representing the specified float value.*

- static **Float valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a **Float** (p. 1328) instance that wraps a primtive float which is parsed from the string value passed.*

Static Public Attributes

- static const int **SIZE** = 32

  The size in bits of the primitive int type.

- static const float **MAX_VALUE**

The maximum value that the primitive type can hold.

- static const float **MIN_VALUE**
  The minimum value that the primitive type can hold.
- static const float **NaN**
  *Constant for the Not a **Number** (p. 1954) Value.*
- static const float **POSITIVE_INFINITY**
  Constant for Positive Infinity.
- static const float **NEGATIVE_INFINITY**
  Constant for Negative Infinity.

6.250.1 Constructor & Destructor Documentation

6.250.1.1 decaf::lang::Float::Float (float *value*)

Parameters:

value - the primitive type to wrap

6.250.1.2 decaf::lang::Float::Float (double *value*)

Parameters:

value - the primitive type to wrap

6.250.1.3 decaf::lang::Float::Float (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the string to convert to a primitive type to wrap

6.250.1.4 virtual decaf::lang::Float::~Float () [inline, virtual]

6.250.2 Member Function Documentation

6.250.2.1 virtual unsigned char decaf::lang::Float::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.250.2.2 static int decaf::lang::Float::compare (float *f1*, float *f2*) [static]

Compares the two specified double values. The sign of the integer value returned is the same as that of the integer that would be returned by the call: Float(f1).compareTo(Float(f2))

Parameters:

f1 - the first double to compare

f2 - the second double to compare

Returns:

the value 0 if d1 is numerically equal to f2; a value less than 0 if f1 is numerically less than f2; and a value greater than 0 if f1 is numerically greater than f2.

6.250.2.3 virtual int decaf::lang::Float::compareTo (const float & *f*) const [virtual]

Compares this **Float** (p. 1328) instance with another.

Parameters:

f - the **Float** (p. 1328) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< float >** (p. 899).

6.250.2.4 virtual int decaf::lang::Float::compareTo (const Float & *f*) const [virtual]

Compares this **Float** (p. 1328) instance with another.

Parameters:

f - the **Float** (p. 1328) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.250.2.5 virtual double decaf::lang::Float::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.250.2.6 bool decaf::lang::Float::equals (const float & *f*) const [inline, virtual]

Parameters:

f - the **Float** (p. 1328) object to compare against.

Returns:

true if the two **Float** (p. 1328) Objects have the same value.

Implements **decaf::lang::Comparable< float >** (p. 900).

6.250.2.7 bool decaf::lang::Float::equals (const Float & *f*) const [inline]

Parameters:

f - the **Float** (p. 1328) object to compare against.

Returns:

true if the two **Float** (p. 1328) Objects have the same value.

6.250.2.8 static int decaf::lang::Float::floatToIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is 0x7fc00000.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1333) method, will produce a floating-point value the same as the argument to floatToIntBits (except all NaN values are collapsed to a single "canonical" NaN value).

Parameters:

value - the float to convert to int bits

Returns:

the int that holds the float's value

6.250.2.9 static int decaf::lang::Float::floatToRawIntBits (float *value*) [static]

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "single format" bit layout, preserving Not-a-Number (NaN) values. Bit 31 (the bit that is selected by the mask 0x80000000) represents the sign of the floating-point number. Bits 30-23 (the bits that are selected by the mask 0x7f800000) represent the exponent. Bits 22-0 (the bits that are selected by the mask 0x007fffff) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7f800000. If the argument is negative infinity, the result is 0xff800000. If the argument is NaN, the result is the integer representing the actual NaN value. Unlike the floatToIntBits method, intToRawIntBits does not collapse all the bit patterns encoding a NaN to a single "canonical" NaN value.

In all cases, the result is an integer that, when given to the **intBitsToFloat(int)** (p. 1333) method, will produce a floating-point value the same as the argument to floatToRawIntBits.

Parameters:

value The float to convert to a raw int.

Returns:

the raw int value of the float

6.250.2.10 virtual float decaf::lang::Float::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.250.2.11 static float decaf::lang::Float::intBitsToFloat (int *bits*) [static]

Returns the float value corresponding to a given bit representation. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "single format" bit layout.

If the argument is 0x7f800000, the result is positive infinity. If the argument is 0xff800000, the result is negative infinity. If the argument is any value in the range 0x7f800001 through 0x7fffffff or in the range 0xff800001 through 0xffffffff, the result is a NaN. No IEEE 754 floating-point operation provided by C++ can distinguish between two NaN values of the same type with different bit patterns. Distinct values of NaN are only distinguishable by use of the **Float::floatToRawIntBits** (p. 1332) method.

Parameters:

bits - the bits of the float encoded as a float

Returns:

a new float created from the int bits.

6.250.2.12 virtual int decaf::lang::Float::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.250.2.13 static bool decaf::lang::Float::isInfinite (float *value*) [static]

Parameters:

value - The float to check.

Returns:

true if the float is equal to infinity.

6.250.2.14 bool decaf::lang::Float::isInfinite () const

Returns:

true if the float is equal to positive infinity.

6.250.2.15 static bool decaf::lang::Float::isNaN (float *value*) [static]

Parameters:

value - The float to check.

Returns:

true if the float is equal to NaN.

6.250.2.16 bool decaf::lang::Float::isNaN () const

Returns:

true if the float is equal to NaN.

6.250.2.17 virtual long long decaf::lang::Float::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.250.2.18 virtual bool decaf::lang::Float::operator< (const float & *f*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< float >** (p. 900).

6.250.2.19 virtual bool decaf::lang::Float::operator< (const Float & *f*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.250.2.20 virtual bool decaf::lang::Float::operator== (const float & *f*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< float >** (p. 901).

6.250.2.21 virtual bool decaf::lang::Float::operator== (const Float & *f*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

f - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.250.2.22 static float decaf::lang::Float::parseFloat (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a new float initialized to the value represented by the specified string, as performed by the valueOf method of class **Float** (p. 1328).

Parameters:

value - the string to parse

Returns:

a float parsed from the string

Exceptions:

NumberFormatException

6.250.2.23 virtual short decaf::lang::Float::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.250.2.24 static std::string decaf::lang::Float::toHexString (float *value*) [static]

Returns a hexadecimal string representation of the float argument. All characters mentioned below are ASCII characters.

∗ If the argument is NaN, the result is the string "NaN". ∗ Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the string "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the string "0x0.0p0"; thus, negative zero produces the result "-0x0.0p0" and positive zero produces the result "0x0.0p0". o If m is a float value with a normalized representation, substrings are used to represent the significand and exponent fields. The significand is represented by the characters "0x1." followed by a lowercase hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed unless all the digits are zero, in which case a single zero is used. Next, the exponent is represented by "p" followed by a decimal string of the unbiased exponent as if produced by a call to **Integer.toString** (p. 1440) on the exponent value. o If m is a float value with a subnormal representation, the significand is represented by the characters "0x0." followed by a hexadecimal representation of the rest of the significand as a fraction. Trailing zeros in the hexadecimal representation are removed. Next, the exponent is represented by "p-126". Note that there must be at least one nonzero digit in a subnormal significand.

Parameters:

value - The float to convert to a string

Returns:

the Hex formatted float string.

6.250.2.25 static std::string decaf::lang::Float::toString (float *value*) [static]

Returns a string representation of the float argument. All characters mentioned below are ASCII characters.

If the argument is NaN, the result is the string "NaN". Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-'; if the sign is positive, no sign character appears in the result. As for the magnitude m: o If m is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity". o If m is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0". o If m is greater than or equal to 10-3 but less than 107, then it is represented as the integer part of m, in decimal form with no leading zeroes, followed by '.', followed by one or more decimal digits representing the fractional part of m. o If m is less than 10-3 or greater than or equal to 107, then it is represented in so-called "computerized scientific notation." Let n be the unique integer such that 10n <= m < 10n+1; then let a be the mathematically exact quotient of m and 10n so that 1 <= a < 10. The magnitude is then represented as the integer part of a, as a single decimal digit, followed by '.', followed by decimal digits representing the fractional part of a, followed by the letter 'E', followed by a representation of n as a decimal integer, as produced by the method **Integer.toString(int)** (p. 1440).

Parameters:

value - The float to convert to a string

Returns:

the formatted float string.

6.250.2.26 std::string decaf::lang::Float::toString () const

Returns:

this **Float** (p. 1328) Object as a String Representation

6.250.2.27 static Float decaf::lang::Float::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Float** (p. 1328) instance that wraps a primtive float which is parsed from the string value passed.

Parameters:

value - the string to parse

Returns:

a new **Float** (p. 1328) instance wrapping the float parsed from value

Exceptions:

NumberFormatException on error.

6.250.2.28 static Float decaf::lang::Float::valueOf (float *value*) [static]

Returns a **Float** (p. 1328) instance representing the specified float value.

Parameters:

value - float to wrap

Returns:

new **Float** (p. 1328) instance wrapping the primitive value

6.250.3 Field Documentation

6.250.3.1 const float decaf::lang::Float::MAX_VALUE [static]

The maximum value that the primitive type can hold.

6.250.3.2 const float decaf::lang::Float::MIN_VALUE [static]

The minimum value that the primitive type can hold.

6.250.3.3 const float decaf::lang::Float::NaN [static]

Constant for the Not a **Number** (p. 1954) Value.

6.250.3.4 const float decaf::lang::Float::NEGATIVE_INFINITY [static]

Constant for Negative Infinity.

6.250.3.5 const float decaf::lang::Float::POSITIVE_INFINITY [static]

Constant for Positive Infinity.

6.250.3.6 const int decaf::lang::Float::SIZE = 32 [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Float.h**

6.251 decaf::internal::nio::FloatArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/FloatArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::FloatArrayBuffer:

Public Member Functions

- **FloatArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **FloatArrayBuffer** (p. 1339) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **FloatArrayBuffer** (float *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **FloatArrayBuffer** (p. 1339) object that wraps the given array.*

- **FloatArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **FloatArrayBuffer** (const **FloatArrayBuffer** &other)

  *Create a **FloatArrayBuffer** (p. 1339) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**FloatArrayBuffer** ()
- virtual float * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the float array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual FloatBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only float buffer that shares this buffer's content.

- virtual FloatBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual FloatBuffer * **duplicate** ()

  Creates a new float buffer that shares this buffer's content.

- virtual float **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual float **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible float array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual FloatBuffer & **put** (float value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual FloatBuffer & **put** (std::size_t index, float value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual FloatBuffer * **slice** () const

  Creates a new FloatBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.251.1 Constructor & Destructor Documentation

6.251.1.1 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **FloatArrayBuffer** (p. 1339) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.251.1.2 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (float * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::NullPointerException)

Creates a **FloatArrayBuffer** (p. 1339) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.251.1.3 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **FloatArrayBuffer** (p. 1339) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.251.1.4 decaf::internal::nio::FloatArrayBuffer::FloatArrayBuffer (const FloatArrayBuffer & *other*)

Create a **FloatArrayBuffer** (p. 1339) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **FloatArrayBuffer** (p. 1339) this one is to mirror.

6.251.1.5 virtual decaf::internal::nio::FloatArrayBuffer::~FloatArrayBuffer () `[virtual]`

6.251.2 Member Function Documentation

6.251.2.1 virtual float* decaf::internal::nio::FloatArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) `[virtual]`

Returns the float array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1348).

6.251.2.2 virtual std::size_t decaf::internal::nio::FloatArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::FloatBuffer** (p. 1349).

6.251.2.3 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only float buffer which the caller then owns.

Implements **decaf::nio::FloatBuffer** (p. 1349).

6.251.2.4 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this FloatBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1349).

6.251.2.5 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::duplicate () [virtual]

Creates a new float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new float Buffer which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1350).

6.251.2.6 virtual float decaf::internal::nio::FloatArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the float is to be read

Returns:

the float that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::FloatBuffer** (p. 1351).

6.251.2.7 virtual float decaf::internal::nio::FloatArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the float at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::FloatBuffer** (p. 1352).

6.251.2.8 virtual bool decaf::internal::nio::FloatArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible float array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::FloatBuffer** (p. 1352).

6.251.2.9 virtual bool decaf::internal::nio::FloatArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.251.2.10 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1353).

6.251.2.11 virtual FloatBuffer& decaf::internal::nio::FloatArrayBuffer::put (float *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::FloatBuffer** (p. 1353).

6.251.2.12 virtual void decaf::internal::nio::FloatArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.251.2.13 virtual FloatBuffer* decaf::internal::nio::FloatArrayBuffer::slice () const [virtual]

Creates a new FloatBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create FloatBuffer which the caller owns.

Implements **decaf::nio::FloatBuffer** (p. 1355).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**FloatArrayBuffer.h**

6.252 decaf::nio::FloatBuffer Class Reference

This class defines four categories of operations upon float buffers:.

`#include <src/main/decaf/nio/FloatBuffer.h>`Inheritance diagram for decaf::nio::FloatBuffer:

Public Member Functions

- virtual ~**FloatBuffer** ()
- virtual std::string **toString** () const
- virtual float * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the float array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **FloatBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only float buffer that shares this buffer's content.

- virtual **FloatBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **FloatBuffer** * **duplicate** ()=0

  Creates a new float buffer that shares this buffer's content.

- virtual float **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual float **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **FloatBuffer** & **get** (std::vector< float > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **FloatBuffer** & **get** (float *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible float array.

- **FloatBuffer** & **put** (**FloatBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the floats remaining in the given source buffer into this buffer.

- **FloatBuffer** & **put** (const float *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers floats into this buffer from the given source array.

- **FloatBuffer** & **put** (std::vector< float > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source floats array into this buffer.

- virtual **FloatBuffer** & **put** (float value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given floats into this buffer at the current position, and then increments the position.

- virtual **FloatBuffer** & **put** (std::size_t index, float value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given floats into this buffer at the given index.

- virtual **FloatBuffer** * **slice** () const =0

  *Creates a new **FloatBuffer** (p. 1346) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **FloatBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **FloatBuffer** &value) const
- virtual bool **operator==** (const **FloatBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **FloatBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **FloatBuffer** * **allocate** (std::size_t capacity)

  Allocates a new Double buffer.

- static **FloatBuffer** * **wrap** (float *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **FloatBuffer** (p. 1346).*

- static **FloatBuffer** * **wrap** (std::vector< float > &buffer)

  *Wraps the passed STL float Vector in a **FloatBuffer** (p. 1346).*

Protected Member Functions

- **FloatBuffer** (std::size_t capacity)

  *Creates a **FloatBuffer** (p. 1346) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.252.1 Detailed Description

This class defines four categories of operations upon float buffers:. o Absolute and relative get and put methods that read and write single floats; o Relative bulk get methods that transfer contiguous sequences of floats from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of floats from a float array or some other float buffer into this buffer o Methods for compacting, duplicating, and slicing a float buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing float array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.252.2 Constructor & Destructor Documentation

6.252.2.1 decaf::nio::FloatBuffer::FloatBuffer (std::size_t *capacity*) [protected]

Creates a **FloatBuffer** (p. 1346) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 627) in doubles

6.252.2.2 virtual decaf::nio::FloatBuffer::~FloatBuffer () [inline, virtual]

6.252.3 Member Function Documentation

6.252.3.1 static FloatBuffer* decaf::nio::FloatBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in floats

Returns:

the **FloatBuffer** (p. 1346) that was allocated, caller owns.

6.252.3.2 virtual float* decaf::nio::FloatBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the float array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1341).

6.252.3.3 virtual std::size_t decaf::nio::FloatBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1342).

6.252.3.4 virtual FloatBuffer* decaf::nio::FloatBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only float buffer which the caller then owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1342).

6.252.3.5 virtual FloatBuffer& decaf::nio::FloatBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **FloatBuffer** (p. 1346)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1342).

6.252.3.6 virtual int decaf::nio::FloatBuffer::compareTo (const FloatBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.252.3.7 virtual FloatBuffer* decaf::nio::FloatBuffer::duplicate () [pure virtual]

Creates a new float buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new float **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1343).

6.252.3.8 virtual bool decaf::nio::FloatBuffer::equals (const FloatBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.252.3.9 FloatBuffer& decaf::nio::FloatBuffer::get (float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers floats from this buffer into the given destination array. If there are fewer floats remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length floats from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length floats remaining in this buffer

NullPointerException if the passed buffer is null.

6.252.3.10 FloatBuffer& decaf::nio::FloatBuffer::get (std::vector< float > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length floats remaining in this buffer

6.252.3.11 virtual float decaf::nio::FloatBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the float is to be read

Returns:

the float that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1343).

6.252.3.12 virtual float decaf::nio::FloatBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the float at the current position

Exceptions:

BufferUnderflowException *(*p. ***661)*** if there no more data to return

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1343).

6.252.3.13 virtual bool decaf::nio::FloatBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible float array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1344).

6.252.3.14 virtual bool decaf::nio::FloatBuffer::operator< (const FloatBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.252.3.15 virtual bool decaf::nio::FloatBuffer::operator== (const FloatBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.252.3.16 virtual FloatBuffer& decaf::nio::FloatBuffer::put (std::size_t *index*, float *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given floats into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the floats to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1344).

6.252.3.17 virtual FloatBuffer& decaf::nio::FloatBuffer::put (float *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given floats into this buffer at the current position, and then increments the position.

Parameters:

value - the floats value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1345).

6.252.3.18 FloatBuffer& decaf::nio::FloatBuffer::put (std::vector< float > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source floats array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **FloatBuffer** (p. 1346)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

6.252.3.19 FloatBuffer& decaf::nio::FloatBuffer::put (const float * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers floats into this buffer from the given source array. If there are more floats to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no floats are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which floats are to be read

offset- The offset within the array of the first float to be read;

length - The number of floats to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.252.3.20 FloatBuffer& decaf::nio::FloatBuffer::put (FloatBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the floats remaining in the given source buffer into this buffer. If there are more floats remaining in the source buffer than in this buffer, that is, if src.remaining() >

remaining() (p. 632), then no floats are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() floats from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take floats from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer for the remaining floats in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.252.3.21 virtual FloatBuffer* decaf::nio::FloatBuffer::slice () const [pure virtual]

Creates a new **FloatBuffer** (p. 1346) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **FloatBuffer** (p. 1346) which the caller owns.

Implemented in **decaf::internal::nio::FloatArrayBuffer** (p. 1345).

6.252.3.22 virtual std::string decaf::nio::FloatBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.252.3.23 static FloatBuffer* decaf::nio::FloatBuffer::wrap (std::vector< float > & *buffer*) [static]

Wraps the passed STL float Vector in a **FloatBuffer** (p. 1346). The new buffer will be backed by the given float array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **FloatBuffer** (p. 1346) that is backed by buffer, caller owns.

6.252.3.24 static FloatBuffer* decaf::nio::FloatBuffer::wrap (float * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **FloatBuffer** (p. 1346). The new buffer will be backed by the given float array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **FloatBuffer** (p. 1346) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**FloatBuffer.h**

6.253 activemq::commands::FlushCommand Class Reference

`#include <src/main/activemq/commands/FlushCommand.h>`Inheritance diagram for activemq::commands::FlushCommand:

Public Member Functions

- **FlushCommand** ()
- virtual ~**FlushCommand** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **FlushCommand** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_FLUSHCOMMAND** = 15

Protected Member Functions

- **FlushCommand** (const **FlushCommand** &)
- **FlushCommand** & **operator**= (const **FlushCommand** &)

6.253.1 Constructor & Destructor Documentation

6.253.1.1 activemq::commands::FlushCommand::FlushCommand (const FlushCommand &) [inline, protected]

6.253.1.2 activemq::commands::FlushCommand::FlushCommand ()

6.253.1.3 virtual activemq::commands::FlushCommand::~FlushCommand () [virtual]

6.253.2 Member Function Documentation

6.253.2.1 virtual FlushCommand* activemq::commands::FlushCommand::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.253.2.2 virtual void activemq::commands::FlushCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.253.2.3 virtual bool activemq::commands::FlushCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.253.2.4 virtual unsigned char activemq::commands::FlushCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.253.2.5 FlushCommand& activemq::commands::FlushCommand::operator= (const FlushCommand &) [inline, protected]

6.253.2.6 virtual std::string activemq::commands::FlushCommand::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.253.2.7 virtual Pointer<Command> activemq::commands::FlushCommand::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.253.3 Field Documentation

6.253.3.1 const unsigned char activemq::commands::FlushCommand::ID_FLUSHCOMMAND = 15 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**FlushCommand.h**

6.254 activemq::wireformat::openwire::marshal::v1::FlushCommandMars Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1360).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/FlushCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.254.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1360). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.254.2 Constructor & Destructor Documentation

6.254.2.1 activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::FlushCommandM () [inline]

6.254.2.2 virtual activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::~FlushCommand () [inline, virtual]

6.254.3 Member Function Documentation

6.254.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.254.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.254.3.3 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.254.3.4 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.254.3.5 virtual int activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.254.3.6 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.254.3.7 virtual void activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**FlushCommandMarshaller.h**

6.255 activemq::wireformat::openwire::marshal::v3::FlushCommandMars Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1364).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/FlushCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.255.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1364). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.255.2 Constructor & Destructor Documentation

6.255.2.1 activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::FlushCommandM () [inline]

6.255.2.2 virtual activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::~FlushCommand () [inline, virtual]

6.255.3 Member Function Documentation

6.255.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.255.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.255.3.3 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.255.3.4 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.255.3.5 virtual int activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.255.3.6 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.255.3.7 virtual void activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**FlushCommandMarshaller.h**

6.256 activemq::wireformat::openwire::marshal::v2::FlushCommandMars Class Reference

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1368).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/FlushCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller:

Public Member Functions

- **FlushCommandMarshaller** ()
- virtual ~**FlushCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.256.1 Detailed Description

Marshaling code for Open Wire Format for **FlushCommandMarshaller** (p. 1368). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.256.2 Constructor & Destructor Documentation

6.256.2.1 activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::FlushCommandM () [inline]

6.256.2.2 virtual activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::~FlushCommand () [inline, virtual]

6.256.3 Member Function Documentation

6.256.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.256.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.256.3.3 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.256.3.4 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.256.3.5 virtual int activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.256.3.6 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.256.3.7 virtual void activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**FlushCommandMarshaller.h**

6.257 decaf::util::logging::Formatter Class Reference

A **Formatter** (p. 1372) provides support for formatting LogRecords.

`#include <src/main/decaf/util/logging/Formatter.h>`Inheritance diagram for decaf::util::logging::Formatter:

Public Member Functions

- virtual ~**Formatter** ()
- virtual std::string **format** (const **LogRecord** &record) const =0

  Format the given log record and return the formatted string.
- virtual std::string **formatMessage** (const **LogRecord** &record) const =0

  Format the message string from a log record.
- virtual std::string **getHead** (const **Handler** *handler)=0

  Return the header string for a set of formatted records.
- virtual std::string **getTail** (const **Handler** *handler)=0

  Return the tail string for a set of formatted records.

6.257.1 Detailed Description

A **Formatter** (p. 1372) provides support for formatting LogRecords. Typically each **logging** (p. 120) **Handler** (p. 1382) will have a **Formatter** (p. 1372) associated with it. The **Formatter** (p. 1372) takes a **LogRecord** (p. 1645) and converts it to a string.

Some formatters (such as the XMLFormatter) need to wrap head and tail strings around a set of formatted records. The getHeader and getTail methods can be used to obtain these strings.

6.257.2 Constructor & Destructor Documentation

6.257.2.1 virtual decaf::util::logging::Formatter::~Formatter () `[inline, virtual]`

6.257.3 Member Function Documentation

6.257.3.1 virtual std::string decaf::util::logging::Formatter::format (const LogRecord & *record*) const `[pure virtual]`

Format the given log record and return the formatted string.

Parameters:

record The Log Record to Format

Returns:

the formatted record.

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2367).

6.257.3.2 virtual std::string decaf::util::logging::Formatter::formatMessage (const LogRecord & *record*) const [pure virtual]

Format the message string from a log record.

Parameters:

record The Log Record to Format

Returns:

the formatted message

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2368).

6.257.3.3 virtual std::string decaf::util::logging::Formatter::getHead (const Handler * *handler*) [pure virtual]

Return the header string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

the head string

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2368).

6.257.3.4 virtual std::string decaf::util::logging::Formatter::getTail (const Handler * *handler*) [pure virtual]

Return the tail string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

the tail string

Implemented in **decaf::util::logging::SimpleFormatter** (p. 2368).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Formatter.h**

6.258 decaf::util::concurrent::Future< V > Class Template Reference

A **Future** (p. 1374) represents the result of an asynchronous computation.

`#include <src/main/decaf/util/concurrent/Future.h>`

Public Member Functions

- virtual ~**Future** ()
- bool **cancel** (bool mayInterruptIfRunning)=0

  Attempts to cancel execution of this task.

- bool **isCancelled** () const =0

  Returns true if this task was canceled before it completed normally.

- bool **isDone** () const =0

  Returns true if this task completed.

- V **get** ()=0 throw (CancellationException, InterruptedException, ExecutionException)

  Waits if necessary for the computation to complete, and then retrieves its result.

- V **get** (long long timeout, **TimeUnit** unit)=0 throw (InterruptedException, ExecutionException, TimeoutException)

  Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

6.258.1 Detailed Description

template<typename V> class decaf::util::concurrent::Future< V >

A **Future** (p. 1374) represents the result of an asynchronous computation. Methods are provided to check if the computation is complete, to wait for its completion, and to retrieve the result of the computation. The result can only be retrieved using method get when the computation has completed, blocking if necessary until it is ready. Cancellation is performed by the cancel method. Additional methods are provided to determine if the task completed normally or was canceled. Once a computation has completed, the computation cannot be canceled. If you would like to use a **Future** (p. 1374) for the sake of cancellability but not provide a usable result, you can declare types of the form Future<void*> and return null as a result of the underlying task.

6.258.2 Constructor & Destructor Documentation

6.258.2.1 template<typename V > virtual decaf::util::concurrent::Future< V >::~Future () [inline, virtual]

6.258.3 Member Function Documentation

6.258.3.1 template<typename V > bool decaf::util::concurrent::Future< V >::cancel (bool *mayInterruptIfRunning*) [pure virtual]

Attempts to cancel execution of this task. This attempt will fail if the task has already completed, has already been canceled, or could not be canceled for some other reason. If successful, and this task has not started when cancel is called, this task should never run. If the task has already started, then the mayInterruptIfRunning parameter determines whether the thread executing this task should be interrupted in an attempt to stop the task.

After this method returns, subsequent calls to **isDone()** (p. 1376) will always return true. Subsequent calls to **isCancelled()** (p. 1376) will always return true if this method returned true.

Parameters:

mayInterruptIfRunning - true if the thread executing this task should be interrupted; otherwise, in-progress tasks are allowed to complete.

Returns:

false if the task could not be canceled, typically because it has already completed normally; true otherwise

6.258.3.2 template<typename V > V decaf::util::concurrent::Future< V >::get (long long *timeout*, TimeUnit *unit*) throw (InterruptedException, ExecutionException, TimeoutException) [pure virtual]

Waits if necessary for at most the given time for the computation to complete, and then retrieves its result, if available.

Parameters:

timeout - the maximum time to wait

unit - the time unit of the timeout argument

Returns:

the computed result

Exceptions:

CancellationException (p. ***784***) - if the computation was canceled

ExecutionException (p. ***1291***) - if the computation threw an exception

InterruptedException - if the current thread was interrupted while waiting

TimeoutException (p. ***2559***) - if the wait timed out

6.258.3.3 template<typename V > V decaf::util::concurrent::Future< V >::get () throw (CancellationException, InterruptedException, ExecutionException) [pure virtual]

Waits if necessary for the computation to complete, and then retrieves its result.

Returns:

the computed result.

Exceptions:

CancellationException *(*p. ***784)*** - if the computation was canceled

ExecutionException *(*p. ***1291)*** - if the computation threw an exception

InterruptedException - if the current thread was interrupted while waiting

6.258.3.4 template<typename V > bool decaf::util::concurrent::Future< V >::isCancelled () const [pure virtual]

Returns true if this task was canceled before it completed normally.

Returns:

true if this task was canceled before it completed

6.258.3.5 template<typename V > bool decaf::util::concurrent::Future< V >::isDone () const [pure virtual]

Returns true if this task completed. Completion may be due to normal termination, an exception, or cancellation -- in all of these cases, this method will return true.

Returns:

true if this task completed

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Future.h**

6.259 activemq::transport::correlator::FutureResponse Class Reference

A container that holds a response object.

`#include <src/main/activemq/transport/correlator/FutureResponse.h>`

Public Member Functions

- **FutureResponse** ()
- virtual ~**FutureResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** () const

  Getters for the response property.

- virtual **Pointer**< **Response** > & **getResponse** ()
- virtual const **Pointer**< **Response** > & **getResponse** (unsigned int timeout) const

  Getters for the response property.

- virtual **Pointer**< **Response** > & **getResponse** (unsigned int timeout)
- virtual void **setResponse** (const **Pointer**< **Response** > &response)

  Setter for the response property.

6.259.1 Detailed Description

A container that holds a response object. Callers of the getResponse method will block until a response has been receive unless they call the getRepsonse that takes a timeout.

6.259.2 Constructor & Destructor Documentation

6.259.2.1 activemq::transport::correlator::FutureResponse::FutureResponse () [inline]

6.259.2.2 virtual activemq::transport::correlator::FutureResponse::~FutureResponse () [inline, virtual]

6.259.3 Member Function Documentation

6.259.3.1 virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int *timeout*) [inline, virtual]

6.259.3.2 virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse (unsigned int *timeout*) const [inline, virtual]

Getters for the response property. Timed Wait.

Parameters:

timeout - time to wait in milliseconds

Returns:

the response object for the request

6.259.3.3 virtual Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () [inline, virtual]

6.259.3.4 virtual const Pointer<Response>& activemq::transport::correlator::FutureResponse::getResponse () const [inline, virtual]

Getters for the response property. Infinite Wait.

Returns:

the response object for the request

6.259.3.5 virtual void activemq::transport::correlator::FutureResponse::setResponse (const Pointer< Response > & *response*) [inline, virtual]

Setter for the response property.

Parameters:

response the response object for the request.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**FutureResponse.h**

6.260 decaf::security::GeneralSecurityException Class Reference

`#include <src/main/decaf/security/GeneralSecurityException.h>`Inheritance diagram for decaf::security::GeneralSecurityException:

Public Member Functions

- **GeneralSecurityException** () throw ()
  Default Constructor.
- **GeneralSecurityException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **GeneralSecurityException** (const **GeneralSecurityException** &ex) throw ()
  Copy Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **GeneralSecurityException** (const std::exception *cause) throw ()
  Constructor.
- **GeneralSecurityException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **GeneralSecurityException** * **clone** () const
  Clones this exception.
- virtual ~**GeneralSecurityException** () throw ()

6.260.1 Constructor & Destructor Documentation

6.260.1.1 decaf::security::GeneralSecurityException::GeneralSecurityException () throw () [inline]

Default Constructor.

6.260.1.2 decaf::security::GeneralSecurityException::GeneralSecurityException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.260.1.3 decaf::security::GeneralSecurityException::GeneralSecurityException (const GeneralSecurityException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.260.1.4 decaf::security::GeneralSecurityException::GeneralSecurityException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.260.1.5 decaf::security::GeneralSecurityException::GeneralSecurityException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.260.1.6 decaf::security::GeneralSecurityException::GeneralSecurityException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.260.1.7 virtual decaf::security::GeneralSecurityException::~GeneralSecurityException () throw () [inline, virtual]

6.260.2 Member Function Documentation

6.260.2.1 virtual GeneralSecurityException* decaf::security::GeneralSecurityException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented in **decaf::security::cert::CertificateEncodingException** (p. 792), **decaf::security::cert::CertificateException** (p. 794), **decaf::security::cert::CertificateExpiredException** (p. 796), **decaf::security::cert::CertificateNotYetValidException** (p. 798), **decaf::security::cert::CertificateParsingException** (p. 800), **decaf::security::InvalidKeyException** (p. 1469), **decaf::security::KeyException** (p. 1574), **decaf::security::NoSuchAlgorithmException** (p. 1944), **decaf::security::NoSuchProviderException** (p. 1950), and **decaf::security::SignatureException** (p. 2366).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**GeneralSecurityException.h**

6.261 decaf::util::logging::Handler Class Reference

A **Handler** (p. 1382) object takes log messages from a **Logger** (p. 1623) and exports them.

`#include <src/main/decaf/util/logging/Handler.h>`Inheritance diagram for decaf::util::logging::Handler:

Public Member Functions

- virtual ~**Handler** ()
- virtual void **flush** ()=0

  Flush the Handler's output, clears any buffers.

- virtual void **publish** (const **LogRecord** &record)=0

  *Publish the Log Record to this **Handler** (p. 1382).*

- virtual void **isLoggable** (const **LogRecord** &record)=0

  *Check if this **Handler** (p. 1382) would actually log a given **LogRecord** (p. 1645).*

- virtual void **setFilter** (const **Filter** *filter)=0

  *Sets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.*

- virtual const **Filter** * **getFilter** ()=0

  *Gets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.*

- virtual void **setLevel** (**Level** value)=0

  ***Set** (p. 2320) the log level specifying which message levels will be logged by this **Handler** (p. 1382).*

- virtual **Level** **getLevel** ()=0

  *Get the log level specifying which message levels will be logged by this **Handler** (p. 1382).*

- virtual void **setFormatter** (const **Formatter** *formatter)=0

  *Sets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).*

- virtual const **Formatter** * **getFormatter** ()=0

  *Gets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).*

6.261.1 Detailed Description

A **Handler** (p. 1382) object takes log messages from a **Logger** (p. 1623) and exports them. It might for example, write them to a console or write them to a file, or send them to a network **logging** (p. 120) service, or forward them to an OS log, or whatever.

A **Handler** (p. 1382) can be disabled by doing a setLevel(Level.OFF) and can be re-enabled by doing a setLevel with an appropriate level.

Handler (p. 1382) classes typically use **LogManager** (p. 1640) properties to set default values for the Handler's **Filter** (p. 1314), **Formatter** (p. 1372), and Level. See the specific documentation for each concrete **Handler** (p. 1382) class.

6.261.2 Constructor & Destructor Documentation

6.261.2.1 virtual decaf::util::logging::Handler::~Handler () [inline, virtual]

6.261.3 Member Function Documentation

6.261.3.1 virtual void decaf::util::logging::Handler::flush () [pure virtual]

Flush the Handler's output, clears any buffers.

Implemented in **decaf::util::logging::StreamHandler** (p. 2473).

6.261.3.2 virtual const Filter* decaf::util::logging::Handler::getFilter () [pure virtual]

Gets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.

Returns:

`Filter` (p. 1314) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2473).

6.261.3.3 virtual const Formatter* decaf::util::logging::Handler::getFormatter () [pure virtual]

Gets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).

Returns:

`Filter` (p. 1314) derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2474).

6.261.3.4 virtual Level decaf::util::logging::Handler::getLevel () [pure virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1382).

Returns:

Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 2474).

6.261.3.5 virtual void decaf::util::logging::Handler::isLoggable (const LogRecord & *record*) [pure virtual]

Check if this **Handler** (p. 1382) would actually log a given **LogRecord** (p. 1645). This method checks if the **LogRecord** (p. 1645) has an appropriate Level and whether it satisfies any **Filter** (p. 1314). It also may make other **Handler** (p. 1382) specific checks that might prevent a handler from **logging** (p. 120) the **LogRecord** (p. 1645).

Parameters:

record `LogRecord (p. 1645)` to check

Implemented in **decaf::util::logging::StreamHandler** (p. 2474).

6.261.3.6 virtual void decaf::util::logging::Handler::publish (const LogRecord & *record*) [pure virtual]

Publish the Log Record to this **Handler** (p. 1382).

Parameters:

record The Log Record to Publish

Implemented in **decaf::util::logging::StreamHandler** (p. 2474).

6.261.3.7 virtual void decaf::util::logging::Handler::setFilter (const Filter * *filter*) [pure virtual]

Sets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records. For each call of publish the **Handler** (p. 1382) will call this **Filter** (p. 1314) (if it is non-null) to check if the **LogRecord** (p. 1645) should be published or discarded.

Parameters:

filter `Filter (p. 1314)` derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2475).

6.261.3.8 virtual void decaf::util::logging::Handler::setFormatter (const Formatter * *formatter*) [pure virtual]

Sets the `Formatter (p. 1372)` used by this **Handler** (p. 1382). Some Handlers may not use Formatters, in which case the **Formatter** (p. 1372) will be remembered, but not used.

Parameters:

formatter `Filter (p. 1314)` derived instance

Implemented in **decaf::util::logging::StreamHandler** (p. 2475).

6.261.3.9 virtual void decaf::util::logging::Handler::setLevel (Level *value*) [pure virtual]

Set (p. 2320) the log level specifying which message levels will be logged by this **Handler** (p. 1382). The intention is to allow developers to turn on voluminous **logging** (p. 120), but to limit the messages that are sent to certain Handlers.

Parameters:

value Level enumeration value

Implemented in **decaf::util::logging::StreamHandler** (p. 2475).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Handler.h**

6.262 decaf::internal::util::HexStringParser Class Reference

`#include <src/main/decaf/internal/util/HexStringParser.h>`

Public Member Functions

- **HexStringParser** (int exponentWidth, int mantissaWidth)

  Create a new HexParser.

- virtual ~**HexStringParser** ()
- long long **parse** (const std::string &hexString)

  Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or doulbe as needed.

Static Public Member Functions

- static double **parseDouble** (const std::string &hexString)
- static float **parseFloat** (const std::string &hexString)

6.262.1 Constructor & Destructor Documentation

6.262.1.1 decaf::internal::util::HexStringParser::HexStringParser (int *exponentWidth*, int *mantissaWidth*)

Create a new HexParser.

Parameters:

exponentWidth - Width of the exponent for the type to parse

mantissaWidth - Width of the mantissa for the type to parse

6.262.1.2 virtual decaf::internal::util::HexStringParser::~HexStringParser () `[inline, virtual]`

6.262.2 Member Function Documentation

6.262.2.1 long long decaf::internal::util::HexStringParser::parse (const std::string & *hexString*)

Parses a hex string using the specs given in the constructor and returns a long long with the bits of the parsed string, the caller can then convert those to a float or doulbe as needed.

Parameters:

hexString - string to parse

Returns:

the bits parsed from the string

6.262.2.2 static double decaf::internal::util::HexStringParser::parseDouble (const std::string & *hexString*) [static]

6.262.2.3 static float decaf::internal::util::HexStringParser::parseFloat (const std::string & *hexString*) [static]

The documentation for this class was generated from the following file:

- src/main/decaf/internal/util/**HexStringParser.h**

6.263 activemq::wireformat::openwire::utils::HexTable Class Reference

The **HexTable** (p. 1388) class maps hexidecimal strings to the value of an index into the table, i.e.

`#include <src/main/activemq/wireformat/openwire/utils/HexTable.h>`

Public Member Functions

- **HexTable** ()
- virtual ~**HexTable** ()
- virtual const std::string & **operator[]** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

- virtual const std::string & **operator[]** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **size** () const

  Returns the max size of this Table.

6.263.1 Detailed Description

The **HexTable** (p. 1388) class maps hexidecimal strings to the value of an index into the table, i.e. the class will return "FF" for the index 255 in the table.

6.263.2 Constructor & Destructor Documentation

6.263.2.1 activemq::wireformat::openwire::utils::HexTable::HexTable ()

6.263.2.2 virtual activemq::wireformat::openwire::utils::HexTable::~HexTable () [inline, virtual]

6.263.3 Member Function Documentation

6.263.3.1 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

6.263.3.2 virtual const std::string& activemq::wireformat::openwire::utils::HexTable::operator[] (std::size_t *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Index operator for this Table, will throw an exception if the index requested is out of bounds for this table.

Parameters:

index The index of the value in the table to fetch.

Returns:

string containing the hex value if the index

Exceptions:

IndexOutOfBoundsException

6.263.3.3 virtual std::size_t activemq::wireformat::openwire::utils::HexTable::size () const [inline, virtual]

Returns the max size of this Table.

Returns:

an integer size value

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**HexTable.h**

6.264 decaf::net::HttpRetryException Class Reference

`#include <src/main/decaf/net/HttpRetryException.h>`Inheritance diagram for decaf::net::HttpRetryException:

Public Member Functions

- **HttpRetryException** () throw ()
  Default Constructor.
- **HttpRetryException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **HttpRetryException** (const **HttpRetryException** &ex) throw ()
  Copy Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **HttpRetryException** (const std::exception ***cause**) throw ()
  Constructor.
- **HttpRetryException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **HttpRetryException** * **clone** () const
  Clones this exception.
- virtual ~**HttpRetryException** () throw ()

6.264.1 Constructor & Destructor Documentation

6.264.1.1 decaf::net::HttpRetryException::HttpRetryException () throw () [inline]

Default Constructor.

6.264.1.2 decaf::net::HttpRetryException::HttpRetryException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.264.1.3 decaf::net::HttpRetryException::HttpRetryException (const HttpRetryException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.264.1.4 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.264.1.5 decaf::net::HttpRetryException::HttpRetryException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.264.1.6 decaf::net::HttpRetryException::HttpRetryException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.264.1.7 virtual decaf::net::HttpRetryException::~HttpRetryException () throw () [inline, virtual]

6.264.2 Member Function Documentation

6.264.2.1 virtual HttpRetryException* decaf::net::HttpRetryException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**HttpRetryException.h**

6.265 decaf::lang::exceptions::IllegalArgumentException Class Reference

`#include <src/main/decaf/lang/exceptions/IllegalArgumentException.h>`Inheritance diagram for decaf::lang::exceptions::IllegalArgumentException:

Public Member Functions

- **IllegalArgumentException** () throw ()
  Default Constructor.
- **IllegalArgumentException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **IllegalArgumentException** (const **IllegalArgumentException** &ex) throw ()
  Copy Constructor.
- **IllegalArgumentException** (const std::exception *__cause__) throw ()
  Constructor.
- **IllegalArgumentException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IllegalArgumentException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **IllegalArgumentException** * **clone** () const
  Clones this exception.
- virtual ~**IllegalArgumentException** () throw ()

6.265.1 Constructor & Destructor Documentation

6.265.1.1 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException () throw () [inline]

Default Constructor.

6.265.1.2 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.265.1.3 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const IllegalArgumentException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.265.1.4 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.265.1.5 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.265.1.6 decaf::lang::exceptions::IllegalArgumentException::IllegalArgumentException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.265.1.7 virtual decaf::lang::exceptions::IllegalArgumentException::~IllegalArgumentException () throw () [inline, virtual]

6.265.2 Member Function Documentation

6.265.2.1 virtual IllegalArgumentException* decaf::lang::exceptions::IllegalArgumentException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalArgumentException.h**

6.266 decaf::lang::exceptions::IllegalMonitorStateException Class Reference

`#include <src/main/decaf/lang/exceptions/IllegalMonitorStateException.h>`Inheritance diagram for decaf::lang::exceptions::IllegalMonitorStateException:

Public Member Functions

- **IllegalMonitorStateException** () throw ()
  Default Constructor.
- **IllegalMonitorStateException** (const **Exception** &ex) throw ()
  Conversion Constructor from some other ***Exception*** *(p. 1268).*
- **IllegalMonitorStateException** (const **IllegalMonitorStateException** &ex) throw ()
  Copy Constructor.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IllegalMonitorStateException** (const std::exception ***cause**) throw ()
  Constructor.
- virtual **IllegalMonitorStateException** * **clone** () const
  Clones this exception.
- virtual ~**IllegalMonitorStateException** () throw ()

6.266.1 Constructor & Destructor Documentation

6.266.1.1 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException () throw () [inline]

Default Constructor.

6.266.1.2 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.266.1.3 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const IllegalMonitorStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.266.1.4 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.266.1.5 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.266.1.6 decaf::lang::exceptions::IllegalMonitorStateException::IllegalMonitorStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.266.1.7 virtual decaf::lang::exceptions::IllegalMonitorStateException::~IllegalMonitorStateException () throw () [inline, virtual]

6.266.2 Member Function Documentation

6.266.2.1 virtual IllegalMonitorStateException* decaf::lang::exceptions::IllegalMonitorStateException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalMonitorStateException.h**

6.267 cms::IllegalStateException Class Reference

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

`#include <src/main/cms/IllegalStateException.h>`Inheritance diagram for cms::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause) throw ()
- **IllegalStateException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**IllegalStateException** () throw ()

6.267.1 Detailed Description

This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation. For example, this exception must be thrown if **Session.commit** (p. 2274) is called on a non-transacted session.

Since:

1.3

6.267.2 Constructor & Destructor Documentation

6.267.2.1 cms::IllegalStateException::IllegalStateException () throw ()

6.267.2.2 cms::IllegalStateException::IllegalStateException (const IllegalStateException & *ex*) throw ()

6.267.2.3 cms::IllegalStateException::IllegalStateException (const std::string & *message*, const std::exception * *cause*) throw ()

6.267.2.4 cms::IllegalStateException::IllegalStateException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.267.2.5 virtual cms::IllegalStateException::~IllegalStateException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**IllegalStateException.h**

6.268 decaf::lang::exceptions::IllegalStateException Class Reference

`#include <src/main/decaf/lang/exceptions/IllegalStateException.h>`Inheritance diagram for decaf::lang::exceptions::IllegalStateException:

Public Member Functions

- **IllegalStateException** () throw ()
  Default Constructor.
- **IllegalStateException** (const **Exception** &ex) throw ()
  Conversion Constructor from some other ***Exception*** *(*p. *1268).*
- **IllegalStateException** (const **IllegalStateException** &ex) throw ()
  Copy Constructor.
- **IllegalStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- **IllegalStateException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IllegalStateException** (const std::exception ***cause**) throw ()
  Constructor.
- virtual **IllegalStateException** * **clone** () const
  Clones this exception.
- virtual ~**IllegalStateException** () throw ()

6.268.1 Constructor & Destructor Documentation

6.268.1.1 decaf::lang::exceptions::IllegalStateException::IllegalStateException () throw () `[inline]`

Default Constructor.

6.268.1.2 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.268.1.3 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const IllegalStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.268.1.4 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.268.1.5 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.268.1.6 decaf::lang::exceptions::IllegalStateException::IllegalStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.268.1.7 virtual decaf::lang::exceptions::IllegalStateException::~IllegalStateException () throw () [inline, virtual]

6.268.2 Member Function Documentation

6.268.2.1 virtual IllegalStateException* decaf::lang::exceptions::IllegalStateException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

Reimplemented in **decaf::nio::InvalidMarkException** (p. 1472).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IllegalStateException.h**

6.269 decaf::lang::exceptions::IndexOutOfBoundsException Class Reference

`#include <src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h>`Inheritance diagram for decaf::lang::exceptions::IndexOutOfBoundsException:

Public Member Functions

- **IndexOutOfBoundsException** () throw ()
  Default Constructor.
- **IndexOutOfBoundsException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **IndexOutOfBoundsException** (const **IndexOutOfBoundsException** &ex) throw ()
  Copy Constructor.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **IndexOutOfBoundsException** (const std::exception ***cause**) throw ()
  Constructor.
- virtual **IndexOutOfBoundsException** * **clone** () const
  Clones this exception.
- virtual ~**IndexOutOfBoundsException** () throw ()

6.269.1 Constructor & Destructor Documentation

6.269.1.1 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException () throw () [inline]

Default Constructor.

6.269.1.2 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.269.1.3 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const IndexOutOfBoundsException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.269.1.4 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.269.1.5 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.269.1.6 decaf::lang::exceptions::IndexOutOfBoundsException::IndexOutOfBoundsException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.269.1.7 virtual decaf::lang::exceptions::IndexOutOfBoundsException::~IndexOutOfBoundsException () throw () [inline, virtual]

6.269.2 Member Function Documentation

6.269.2.1 virtual IndexOutOfBoundsException* decaf::lang::exceptions::IndexOutOfBoundsException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**IndexOutOfBoundsException.h**

6.270 decaf::io::InputStream Class Reference

Base interface for an input stream.

`#include <src/main/decaf/io/InputStream.h>`Inheritance diagram for decaf::io::InputStream:

Public Member Functions

- virtual ~**InputStream** ()
- virtual void **mark** (int readLimit)=0

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** ()=0 throw (IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const =0

  Determines if this input stream supports the mark and reset methods.

- virtual std::size_t **available** () const =0 throw (IOException)

  Indicates the number of bytes available.

- virtual unsigned char **read** ()=0 throw (IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize)=0 throw (IOException, lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual std::size_t **skip** (std::size_t num)=0 throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

6.270.1 Detailed Description

Base interface for an input stream.

6.270.2 Constructor & Destructor Documentation

6.270.2.1 virtual decaf::io::InputStream::~InputStream () [inline, virtual]

6.270.3 Member Function Documentation

6.270.3.1 virtual std::size_t decaf::io::InputStream::available () const throw (IOException) [pure virtual]

Indicates the number of bytes available.

Returns:

the number of bytes available on this input stream.

Exceptions:

IOException *(p. 1477)* if an error occurs.

Implemented in **decaf::internal::io::StandardInputStream** (p. 2404), **decaf::io::BlockingByteArrayInputStream** (p. 568), **decaf::io::BufferedInputStream** (p. 634), **decaf::io::ByteArrayInputStream** (p. 718), **decaf::io::FilterInputStream** (p. 1317), and **decaf::net::SocketInputStream** (p. 2385).

6.270.3.2 virtual void decaf::io::InputStream::mark (int *readLimit*) [pure virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implemented in **decaf::internal::io::StandardInputStream** (p. 2405), **decaf::io::BlockingByteArrayInputStream** (p. 568), **decaf::io::BufferedInputStream** (p. 635), **decaf::io::ByteArrayInputStream** (p. 718), **decaf::io::FilterInputStream** (p. 1318), and **decaf::net::SocketInputStream** (p. 2386).

6.270.3.3 virtual bool decaf::io::InputStream::markSupported () const [pure virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implemented in **decaf::internal::io::StandardInputStream** (p. 2405), **decaf::io::BlockingByteArrayInputStream** (p. 569), **decaf::io::BufferedInputStream** (p. 635), **decaf::io::ByteArrayInputStream** (p. 719), **decaf::io::FilterInputStream** (p. 1318), and **decaf::net::SocketInputStream** (p. 2386).

6.270.3.4 virtual int decaf::io::InputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

NullPointerException if buffer is null

Implemented in **activemq::io::LoggingInputStream** (p. 1633), **decaf::internal::io::StandardInputStream** (p. 2406), **decaf::io::BlockingByteArrayInputStream** (p. 569), **decaf::io::BufferedInputStream** (p. 635), **decaf::io::ByteArrayInputStream** (p. 719), **decaf::io::DataInputStream** (p. 1118), **decaf::io::FilterInputStream** (p. 1319), and **decaf::net::SocketInputStream** (p. 2387).

6.270.3.5 virtual unsigned char decaf::io::InputStream::read () throw (IOException) [pure virtual]

Reads a single byte from the buffer. Blocks until data is available.

Returns:

The next byte.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implemented in **activemq::io::LoggingInputStream** (p. 1634), **decaf::internal::io::StandardInputStream** (p. 2406), **decaf::io::BlockingByteArrayInputStream** (p. 570), **decaf::io::BufferedInputStream** (p. 636), **decaf::io::ByteArrayInputStream** (p. 720), **decaf::io::FilterInputStream** (p. 1319), and **decaf::net::SocketInputStream** (p. 2387).

6.270.3.6 virtual void decaf::io::InputStream::reset () throw (IOException) [pure virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an **IOException** (p. 1477)

might be thrown. * If such an **IOException** (p. 1477) is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an **IOException** (p. 1477). * If an **IOException** (p. 1477) is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException *(p. 1477)*

Implemented in **decaf::internal::io::StandardInputStream** (p. 2406), **decaf::io::BlockingByteArrayInputStream** (p. 570), **decaf::io::BufferedInputStream** (p. 636), **decaf::io::ByteArrayInputStream** (p. 720), **decaf::io::FilterInputStream** (p. 1320), and **decaf::net::SocketInputStream** (p. 2388).

6.270.3.7 virtual std::size_t decaf::io::InputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of **InputStream** (p. 1406) creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

IOException *(p. 1477)* if an error occurs

Implemented in **decaf::internal::io::StandardInputStream** (p. 2407), **decaf::io::BlockingByteArrayInputStream** (p. 571), **decaf::io::BufferedInputStream** (p. 636), **decaf::io::ByteArrayInputStream** (p. 721), **decaf::io::DataInputStream** (p. 1124), **decaf::io::FilterInputStream** (p. 1320), and **decaf::net::SocketInputStream** (p. 2388).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InputStream.h**

6.271 decaf::internal::nio::IntArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/IntArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::IntArrayBuffer:

Public Member Functions

- **IntArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **IntArrayBuffer** (p. 1410) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **IntArrayBuffer** (int *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **IntArrayBuffer** (p. 1410) object that wraps the given array.*

- **IntArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **IntArrayBuffer** (const **IntArrayBuffer** &other)

  *Create a **IntArrayBuffer** (p. 1410) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**IntArrayBuffer** ()
- virtual int * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the int array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual IntBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only int buffer that shares this buffer's content.

- virtual IntBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual IntBuffer * **duplicate** ()

  Creates a new int buffer that shares this buffer's content.

- virtual int **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual int **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible int array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual IntBuffer & **put** (int value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual IntBuffer & **put** (std::size_t index, int value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual IntBuffer * **slice** () const

  Creates a new IntBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.271.1 Constructor & Destructor Documentation

6.271.1.1 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **IntArrayBuffer** (p. 1410) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.271.1.2 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (int * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::NullPointerException)

Creates a **IntArrayBuffer** (p. 1410) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.271.1.3 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **IntArrayBuffer** (p. 1410) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.271.1.4 decaf::internal::nio::IntArrayBuffer::IntArrayBuffer (const IntArrayBuffer & *other*)

Create a **IntArrayBuffer** (p. 1410) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **IntArrayBuffer** (p. 1410) this one is to mirror.

6.271.1.5 virtual decaf::internal::nio::IntArrayBuffer::~IntArrayBuffer () `[virtual]`

6.271.2 Member Function Documentation

6.271.2.1 virtual int* decaf::internal::nio::IntArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) `[virtual]`

Returns the int array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1419).

6.271.2.2 virtual std::size_t decaf::internal::nio::IntArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::IntBuffer** (p. 1420).

6.271.2.3 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only int buffer which the caller then owns.

Implements **decaf::nio::IntBuffer** (p. 1420).

6.271.2.4 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this IntBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1420).

6.271.2.5 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::duplicate () [virtual]

Creates a new int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new int Buffer which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1421).

6.271.2.6 virtual int decaf::internal::nio::IntArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the int is to be read

Returns:

the int that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::IntBuffer** (p. 1422).

6.271.2.7 virtual int decaf::internal::nio::IntArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the int at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::IntBuffer** (p. 1423).

6.271.2.8 virtual bool decaf::internal::nio::IntArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible int array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::IntBuffer** (p. 1423).

6.271.2.9 virtual bool decaf::internal::nio::IntArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.271.2.10 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1424).

6.271.2.11 virtual IntBuffer& decaf::internal::nio::IntArrayBuffer::put (int *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::IntBuffer** (p. 1424).

6.271.2.12 virtual void decaf::internal::nio::IntArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.271.2.13 virtual IntBuffer* decaf::internal::nio::IntArrayBuffer::slice () const [virtual]

Creates a new IntBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create IntBuffer which the caller owns.

Implements **decaf::nio::IntBuffer** (p. 1426).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**IntArrayBuffer.h**

6.272 decaf::nio::IntBuffer Class Reference

This class defines four categories of operations upon int buffers:.

`#include <src/main/decaf/nio/IntBuffer.h>`Inheritance diagram for decaf::nio::IntBuffer:

Public Member Functions

- virtual ~**IntBuffer** ()
- virtual std::string **toString** () const
- virtual int * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the int array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **IntBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only int buffer that shares this buffer's content.

- virtual **IntBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **IntBuffer** * **duplicate** ()=0

  Creates a new int buffer that shares this buffer's content.

- virtual int **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual int **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **IntBuffer** & **get** (std::vector< int > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **IntBuffer** & **get** (int *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible int array.

- **IntBuffer** & **put** (**IntBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the ints remaining in the given source buffer into this buffer.

- **IntBuffer** & **put** (const int *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers ints into this buffer from the given source array.

- **IntBuffer** & **put** (std::vector< int > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source ints array into this buffer.

- virtual **IntBuffer** & **put** (int value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given ints into this buffer at the current position, and then increments the position.

- virtual **IntBuffer** & **put** (std::size_t index, int value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given ints into this buffer at the given index.

- virtual **IntBuffer** * **slice** () const =0

  *Creates a new **IntBuffer** (p. 1417) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **IntBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **IntBuffer** &value) const
- virtual bool **operator==** (const **IntBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **IntBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **IntBuffer** * **allocate** (std::size_t capacity)

  Allocates a new Double buffer.

- static **IntBuffer** * **wrap** (int *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **IntBuffer** (p. 1417).*

- static **IntBuffer** * **wrap** (std::vector< int > &buffer)

  *Wraps the passed STL int Vector in a **IntBuffer** (p. 1417).*

Protected Member Functions

- **IntBuffer** (std::size_t capacity)

  *Creates a **IntBuffer** (p. 1417) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.272.1 Detailed Description

This class defines four categories of operations upon int buffers:. o Absolute and relative get and put methods that read and write single ints; o Relative bulk get methods that transfer contiguous sequences of ints from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of ints from a int array or some other int buffer into this buffer o Methods for compacting, duplicating, and slicing a int buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing int array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.272.2 Constructor & Destructor Documentation

6.272.2.1 decaf::nio::IntBuffer::IntBuffer (std::size_t *capacity*) [protected]

Creates a **IntBuffer** (p. 1417) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 627) in doubles

6.272.2.2 virtual decaf::nio::IntBuffer::~IntBuffer () [inline, virtual]

6.272.3 Member Function Documentation

6.272.3.1 static IntBuffer* decaf::nio::IntBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in ints

Returns:

the **IntBuffer** (p. 1417) that was allocated, caller owns.

6.272.3.2 virtual int* decaf::nio::IntBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the int array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1412).

6.272.3.3 virtual std::size_t decaf::nio::IntBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.

UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1413).

6.272.3.4 virtual IntBuffer* decaf::nio::IntBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only int buffer which the caller then owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1413).

6.272.3.5 virtual IntBuffer& decaf::nio::IntBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **IntBuffer** (p. 1417)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1413).

6.272.3.6 virtual int decaf::nio::IntBuffer::compareTo (const IntBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.272.3.7 virtual IntBuffer* decaf::nio::IntBuffer::duplicate () [pure virtual]

Creates a new int buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new int **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1414).

6.272.3.8 virtual bool decaf::nio::IntBuffer::equals (const IntBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.272.3.9 IntBuffer& decaf::nio::IntBuffer::get (int * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers ints from this buffer into the given destination array. If there are fewer ints remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length ints from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length ints remaining in this buffer

NullPointerException if the passed buffer is null.

6.272.3.10 IntBuffer& decaf::nio::IntBuffer::get (std::vector< int > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(p. **661**)* - If there are fewer than length ints remaining in this buffer

6.272.3.11 virtual int decaf::nio::IntBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the int is to be read

Returns:

the int that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1414).

6.272.3.12 virtual int decaf::nio::IntBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the int at the current position

Exceptions:

BufferUnderflowException *(*p. ***661)*** if there no more data to return

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1414).

6.272.3.13 virtual bool decaf::nio::IntBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible int array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1415).

6.272.3.14 virtual bool decaf::nio::IntBuffer::operator< (const IntBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.272.3.15 virtual bool decaf::nio::IntBuffer::operator== (const IntBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.272.3.16 virtual IntBuffer& decaf::nio::IntBuffer::put (std::size_t *index*, int *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given ints into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the ints to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1415).

6.272.3.17 virtual IntBuffer& decaf::nio::IntBuffer::put (int *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given ints into this buffer at the current position, and then increments the position.

Parameters:

value - the ints value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1416).

6.272.3.18 IntBuffer& decaf::nio::IntBuffer::put (std::vector< int > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source ints array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **IntBuffer** (p. 1417)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

6.272.3.19 IntBuffer& decaf::nio::IntBuffer::put (const int * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers ints into this buffer from the given source array. If there are more ints to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no ints are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which ints are to be read

offset- The offset within the array of the first int to be read;

length - The number of ints to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException (p. ***658***) - If there is insufficient space in this buffer

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.272.3.20 IntBuffer& decaf::nio::IntBuffer::put (IntBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the ints remaining in the given source buffer into this buffer. If there are more ints remaining in the source buffer than in this buffer, that is, if src.remaining() >

remaining() (p. 632), then no ints are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() ints from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take ints from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658)*** - If there is insufficient space in this buffer for the remaining ints in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(*p. ***2167)*** - If this buffer is read-only

6.272.3.21 virtual IntBuffer* decaf::nio::IntBuffer::slice () const [pure virtual]

Creates a new **IntBuffer** (p. 1417) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **IntBuffer** (p. 1417) which the caller owns.

Implemented in **decaf::internal::nio::IntArrayBuffer** (p. 1416).

6.272.3.22 virtual std::string decaf::nio::IntBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.272.3.23 static IntBuffer* decaf::nio::IntBuffer::wrap (std::vector< int > & *buffer*) [static]

Wraps the passed STL int Vector in a **IntBuffer** (p. 1417). The new buffer will be backed by the given int array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **IntBuffer** (p. 1417) that is backed by buffer, caller owns.

6.272.3.24 static IntBuffer* decaf::nio::IntBuffer::wrap (int * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **IntBuffer** (p. 1417). The new buffer will be backed by the given int array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **IntBuffer** (p. 1417) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**IntBuffer.h**

6.273 decaf::lang::Integer Class Reference

`#include <src/main/decaf/lang/Integer.h>`Inheritance diagram for decaf::lang::Integer:

Public Member Functions

- **Integer** (int value)
- **Integer** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Integer** ()
- virtual int **compareTo** (const **Integer** &i) const

  *Compares this **Integer** (p. 1428) instance with another.*

- bool **equals** (const **Integer** &i) const
- virtual bool **operator==** (const **Integer** &i) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Integer** &i) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const int &i) const

  *Compares this **Integer** (p. 1428) instance with another.*

- bool **equals** (const int &i) const
- virtual bool **operator==** (const int &i) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const int &i) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

Static Public Member Functions

- static **Integer decode** (const std::string &value) throw (exceptions::NumberFormatException)

  *Decodes a String into a **Integer** (p. 1428).*

- static int **reverseBytes** (int value)

  Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

- static int **reverse** (int value)

  Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

- static int **parseInt** (const std::string &s, int radix) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed int in the radix specified by the second argument.

- static int **parseInt** (const std::string &s) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed decimal int.

- static **Integer valueOf** (int value)

  *Returns a **Integer** (p. 1428) instance representing the specified int value.*

- static **Integer valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a **Integer** (p. 1428) object holding the value given by the specified std::string.*

- static **Integer valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

  *Returns a **Integer** (p. 1428) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

- static int **bitCount** (int value)

  Returns the number of one-bits in the two's complement binary representation of the specified int value.

- static std::string **toString** (int value)

  Converts the int to a String representation.

- static std::string **toString** (int value, int radix)

  Returns a string representation of the first argument in the radix specified by the second argument.

- static std::string **toHexString** (int value)

  Returns a string representation of the integer argument as an unsigned integer in base 16.

- static std::string **toOctalString** (int value)

  Returns a string representation of the integer argument as an unsigned integer in base 8.

- static std::string **toBinaryString** (int value)

  Returns a string representation of the integer argument as an unsigned integer in base 2.

- static int **highestOneBit** (int value)

  Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

- static int **lowestOneBit** (int value)

  Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

- static int **numberOfLeadingZeros** (int value)

  Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value.

- static int **numberOfTrailingZeros** (int value)

  Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value.

- static int **rotateLeft** (int value, int distance)

  Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

- static int **rotateRight** (int value, int distance)

  Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

- static int **signum** (int value)

  Returns the signum function of the specified int value.

Static Public Attributes

- static const int **SIZE** = 32

  The size in bits of the primitive int type.

- static const int **MAX_VALUE** = (int)0x7FFFFFFF

  The maximum value that the primitive type can hold.

- static const int **MIN_VALUE** = (int)0x80000000

  The minimum value that the primitive type can hold.

6.273.1 Constructor & Destructor Documentation

6.273.1.1 decaf::lang::Integer::Integer (int *value*)

Parameters:

value The primitive value to wrap in an `Integer` (p. 1428) instance.

6.273.1.2 decaf::lang::Integer::Integer (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value The base 10 encoded string to decode to an `Integer` (p. 1428) and wrap.

Exceptions:

NumberFormatException

6.273.1.3 virtual decaf::lang::Integer::~Integer () [inline, virtual]

6.273.2 Member Function Documentation

6.273.2.1 static int decaf::lang::Integer::bitCount (int *value*) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

Parameters:

value - the int to count

Returns:

the number of one-bits in the two's complement binary representation of the specified int value.

6.273.2.2 virtual unsigned char decaf::lang::Integer::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.273.2.3 virtual int decaf::lang::Integer::compareTo (const int & *i*) const [virtual]

Compares this **Integer** (p. 1428) instance with another.

Parameters:

i - the **Integer** (p. 1428) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

Implements **decaf::lang::Comparable< int >** (p. 899).

6.273.2.4 virtual int decaf::lang::Integer::compareTo (const Integer & *i*) const [virtual]

Compares this **Integer** (p. 1428) instance with another.

Parameters:

i - the **Integer** (p. 1428) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object represents a value less than the passed in value.

6.273.2.5 static Integer decaf::lang::Integer::decode (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Integer** (p. 1428). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInteger method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Integer** (p. 1428) object containing the decoded value

Exceptions:

NumberFomatException if the string is not formatted correctly.

6.273.2.6 virtual double decaf::lang::Integer::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.273.2.7 bool decaf::lang::Integer::equals (const int & *i*) const [inline, virtual]

Parameters:

i - the **Integer** (p. 1428) object to compare against.

Returns:

true if the two **Integer** (p. 1428) Objects have the same value.

Implements **decaf::lang::Comparable< int >** (p. 900).

6.273.2.8 bool decaf::lang::Integer::equals (const Integer & *i*) const [inline]

Parameters:

i - the **Integer** (p. 1428) object to compare against.

Returns:

true if the two **Integer** (p. 1428) Objects have the same value.

6.273.2.9 virtual float decaf::lang::Integer::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.273.2.10 static int decaf::lang::Integer::highestOneBit (int *value*) [static]

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.273.2.11 virtual int decaf::lang::Integer::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.273.2.12 virtual long long decaf::lang::Integer::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.273.2.13 static int decaf::lang::Integer::lowestOneBit (int *value*) [static]

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.273.2.14 static int decaf::lang::Integer::numberOfLeadingZeros (int *value*) [static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

∗ floor(log2(x)) = 31 - numberOfLeadingZeros(x) ∗ ceil(log2(x)) = 32 - numberOfLeadingZeros(x - 1)

Parameters:

value - the int to be inspected

Returns:

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.273.2.15 static int decaf::lang::Integer::numberOfTrailingZeros (int *value*) [static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.

6.273.2.16 virtual bool decaf::lang::Integer::operator< (const int & *i*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 900).

6.273.2.17 virtual bool decaf::lang::Integer::operator< (const Integer & *i*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.273.2.18 virtual bool decaf::lang::Integer::operator== (const int & *i*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< int >** (p. 901).

6.273.2.19 virtual bool decaf::lang::Integer::operator== (const Integer & *i*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

i - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.273.2.20 static int decaf::lang::Integer::parseInt (const std::string & *s*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal int. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting int value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseInteger(const std::string, int) method.

Parameters:

s - String to convert to a int

Returns:

the converted int value

Exceptions:

NumberFormatException if the string is not a int.

6.273.2.21 static int decaf::lang::Integer::parseInt (const std::string & *s*, int *radix*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed int in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 804) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type NumberFormatException is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 808) or larger than **Character.MAX_RADIX** (p. 807). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type int.

Parameters:

s - the String containing the int representation to be parsed

radix - the radix to be used while parsing s

Returns:

the int represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable int.

6.273.2.22 static int decaf::lang::Integer::reverse (int *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

Parameters:

value - the value whose bits are to be reversed

Returns:

the reversed bits int.

6.273.2.23 static int decaf::lang::Integer::reverseBytes (int *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

Parameters:

value - the int whose bytes we are to reverse

Returns:

the reversed int.

6.273.2.24 static int decaf::lang::Integer::rotateLeft (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters:

value - the int to be inspected

distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.

6.273.2.25 static int decaf::lang::Integer::rotateRight (int *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

Parameters:

value - the int to be inspected

distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.

6.273.2.26 virtual short decaf::lang::Integer::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.273.2.27 static int decaf::lang::Integer::signum (int *value*) [static]

Returns the signum function of the specified int value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters:

value - the int to be inspected

Returns:

the signum function of the specified int value.

6.273.2.28 static std::string decaf::lang::Integer::toBinaryString (int *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 2. The unsigned integer value is the argument plus 2^32 if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character.

The characters '0' and '1' are used as binary digits.

Parameters:

value - the int to be translated to a binary string

Returns:

the unsigned int value as a binary string

6.273.2.29 static std::string decaf::lang::Integer::toHexString (int *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus 2^32 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters:

value - the int to be translated to an Octal string

Returns:

the unsigned int value as a Octal string

6.273.2.30 static std::string decaf::lang::Integer::toOctalString (int *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 8. The unsigned integer value is the argument plus 2^32 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters:

value - the int to be translated to an Octal string

Returns:

the unsigned int value as a Octal string

6.273.2.31 static std::string decaf::lang::Integer::toString (int *value*, int *radix*) [static]

Returns a string representation of the first argument in the radix specified by the second argument. If the radix is smaller than **Character.MIN_RADIX** (p. 808) or larger than **Character.MAX_RADIX** (p. 807), then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-'. If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

Parameters:

value - the int to convert to a string

radix - the radix to format the string in

Returns:

an int formatted to the string value of the radix given.

6.273.2.32 static std::string decaf::lang::Integer::toString (int *value*) [static]

Converts the int to a String representation.

Parameters:

value The int to convert to a `std::string` instance.

Returns:

string representation

6.273.2.33 std::string decaf::lang::Integer::toString () const

Returns:

this `Integer` (p. 1428) Object as a String Representation

6.273.2.34 static Integer decaf::lang::Integer::valueOf (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Integer** (p. 1428) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed int in the radix specified by the second argument, exactly as if the argument were given to the parseInt(std::string, int) method. The result is a **Integer** (p. 1428) object that represents the int value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Integer** (p. 1428) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid int.

6.273.2.35 static Integer decaf::lang::Integer::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Integer** (p. 1428) object holding the value given by the specified std::string. The argument is interpreted as representing a signed decimal int, exactly as if the argument were given to the parseInt(std::string) method. The result is a **Integer** (p. 1428) object that represents the int value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Integer** (p. 1428) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal int.

6.273.2.36 static Integer decaf::lang::Integer::valueOf (int *value*) [inline, static]

Returns a **Integer** (p. 1428) instance representing the specified int value.

Parameters:

value - the int to wrap

Returns:

the new **Integer** (p. 1428) object wrapping value.

6.273.3 Field Documentation

6.273.3.1 const int decaf::lang::Integer::MAX_VALUE = (int)0x7FFFFFFF [static]

The maximum value that the primitive type can hold.

6.273.3.2 const int decaf::lang::Integer::MIN_VALUE = (int)0x80000000 [static]

The minimum value that the primitive type can hold.

6.273.3.3 const int decaf::lang::Integer::SIZE = 32 [static]

The size in bits of the primitive int type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Integer.h**

6.274 activemq::commands::IntegerResponse Class Reference

`#include <src/main/activemq/commands/IntegerResponse.h>`Inheritance diagram for activemq::commands::IntegerResponse:

Public Member Functions

- **IntegerResponse** ()
- virtual ~**IntegerResponse** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **IntegerResponse** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual int **getResult** () const
- virtual void **setResult** (int **result**)

Static Public Attributes

- static const unsigned char **ID_INTEGERRESPONSE** = 34

Protected Member Functions

- **IntegerResponse** (const **IntegerResponse** &)
- **IntegerResponse** & **operator**= (const **IntegerResponse** &)

Protected Attributes

- int **result**

6.274.1 Constructor & Destructor Documentation

6.274.1.1 activemq::commands::IntegerResponse::IntegerResponse (const IntegerResponse &) [inline, protected]

6.274.1.2 activemq::commands::IntegerResponse::IntegerResponse ()

6.274.1.3 virtual activemq::commands::IntegerResponse::~IntegerResponse () [virtual]

6.274.2 Member Function Documentation

6.274.2.1 virtual IntegerResponse* activemq::commands::IntegerResponse::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.274.2.2 virtual void activemq::commands::IntegerResponse::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::Response** (p. 2232).

6.274.2.3 virtual bool activemq::commands::IntegerResponse::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::Response** (p. 2232).

6.274.2.4 virtual unsigned char activemq::commands::IntegerResponse::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.274.2.5 virtual int activemq::commands::IntegerResponse::getResult () const [virtual]

6.274.2.6 IntegerResponse& activemq::commands::IntegerResponse::operator= (const IntegerResponse &) [inline, protected]

Reimplemented from **activemq::commands::Response** (p. 2233).

6.274.2.7 virtual void activemq::commands::IntegerResponse::setResult (int *result*) [virtual]

6.274.2.8 virtual std::string activemq::commands::IntegerResponse::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::Response** (p. 2233).

6.274.3 Field Documentation

6.274.3.1 const unsigned char activemq::commands::IntegerResponse::ID_INTEGERRESPONSE = 34 [static]

6.274.3.2 int activemq::commands::IntegerResponse::result [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**IntegerResponse.h**

6.275 activemq::wireformat::openwire::marshal::v2::IntegerResponseMar Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1445).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/IntegerResponseMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual ~**IntegerResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.275.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1445). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.275.2 Constructor & Destructor Documentation

6.275.2.1 activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::IntegerResponse () [inline]

6.275.2.2 virtual activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::~IntegerRespon () [inline, virtual]

6.275.3 Member Function Documentation

6.275.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.275.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.275.3.3 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2242).

6.275.3.4 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.275.3.5 virtual int activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2243).

6.275.3.6 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2244).

6.275.3.7 virtual void activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller** (p. 2245).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**IntegerResponseMarshaller.h**

6.276 activemq::wireformat::openwire::marshal::v1::IntegerResponseMar Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1449).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual ~**IntegerResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.276.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1449). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.276.2 Constructor & Destructor Documentation

6.276.2.1 activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::IntegerResponse () [inline]

6.276.2.2 virtual activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::~IntegerRespon () [inline, virtual]

6.276.3 Member Function Documentation

6.276.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.276.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.276.3.3 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2252).

6.276.3.4 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.276.3.5 virtual int activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2253).

6.276.3.6 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2254).

6.276.3.7 virtual void activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller** (p. 2255).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**IntegerResponseMarshaller.h**

6.277 activemq::wireformat::openwire::marshal::v3::IntegerResponseMar Class Reference

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1453).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/IntegerResponseMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller:

Public Member Functions

- **IntegerResponseMarshaller** ()
- virtual ~**IntegerResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.277.1 Detailed Description

Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1453). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.277.2 Constructor & Destructor Documentation

6.277.2.1 activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::IntegerResponse () [inline]

6.277.2.2 virtual activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::~IntegerRespon () [inline, virtual]

6.277.3 Member Function Documentation

6.277.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.277.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.277.3.3 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2247).

6.277.3.4 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.277.3.5 virtual int activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2248).

6.277.3.6 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2249).

6.277.3.7 virtual void activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller** (p. 2250).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**IntegerResponseMarshaller.h**

6.278 activemq::transport::mock::InternalCommandListener Class Reference

Listens for Commands sent from the **MockTransport** (p. 1910).

`#include <src/main/activemq/transport/mock/InternalCommandListener.h>`Inheritance diagram for activemq::transport::mock::InternalCommandListener:

Public Member Functions

- **InternalCommandListener** ()
- virtual ~**InternalCommandListener** ()
- void **setTransport** (**MockTransport** *transport)
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

  Event handler for the receipt of a command.

- void **run** ()

  Default implementation of the run method - does nothing.

6.278.1 Detailed Description

Listens for Commands sent from the **MockTransport** (p. 1910). This class processes all outbound **commands** (p. 59) and sends responses that are constructed by calling the Protocol provided **ResponseBuilder** (p. 2235) and getting a set of Commands to send back into the **MockTransport** (p. 1910) as incoming Commands and Responses.

6.278.2 Constructor & Destructor Documentation

6.278.2.1 activemq::transport::mock::InternalCommandListener::InternalCommandListener ()

6.278.2.2 virtual activemq::transport::mock::InternalCommandListener::~InternalCommandListener () `[virtual]`

6.278.3 Member Function Documentation

6.278.3.1 virtual void activemq::transport::mock::InternalCommandListener::onCommand (const Pointer< Command > & *command*) `[virtual]`

Event handler for the receipt of a command. The **transport** (p. 67) passes off all received **commands** (p. 59) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2608) deletes the command upon receipt.

Parameters:

command the received command object.

Implements **activemq::transport::TransportListener** (p. 2624).

6.278.3.2 void activemq::transport::mock::InternalCommandListener::run () [virtual]

Default implementation of the run method - does nothing.

Reimplemented from **decaf::lang::Thread** (p. 2545).

6.278.3.3 void activemq::transport::mock::InternalCommandListener::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

6.278.3.4 void activemq::transport::mock::InternalCommandListener::setTransport (MockTransport * *transport*) [inline]

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**InternalCommandListener.h**

6.279 decaf::lang::exceptions::InterruptedException Class Reference

`#include <src/main/decaf/lang/exceptions/InterruptedException.h>`Inheritance diagram for decaf::lang::exceptions::InterruptedException:

Public Member Functions

- **InterruptedException** () throw ()
  Default Constructor.
- **InterruptedException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **InterruptedException** (const **InterruptedException** &ex) throw ()
  Copy Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedException** (const std::exception ***cause**) throw ()
  Constructor.
- **InterruptedException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **InterruptedException** * **clone** () const
  Clones this exception.
- virtual ~**InterruptedException** () throw ()

6.279.1 Constructor & Destructor Documentation

6.279.1.1 decaf::lang::exceptions::InterruptedException::InterruptedException () throw () [inline]

Default Constructor.

6.279.1.2 decaf::lang::exceptions::InterruptedException::InterruptedException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.279.1.3 decaf::lang::exceptions::InterruptedException::InterruptedException (const InterruptedException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.279.1.4 decaf::lang::exceptions::InterruptedException::InterruptedException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.279.1.5 decaf::lang::exceptions::InterruptedException::InterruptedException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.279.1.6 decaf::lang::exceptions::InterruptedException::InterruptedException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.279.1.7 virtual decaf::lang::exceptions::InterruptedException::~InterruptedException () throw () [inline, virtual]

6.279.2 Member Function Documentation

6.279.2.1 virtual InterruptedException* decaf::lang::exceptions::InterruptedException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InterruptedException.h**

6.280 decaf::io::InterruptedIOException Class Reference

`#include <src/main/decaf/io/InterruptedIOException.h>`Inheritance diagram for decaf::io::InterruptedIOException:

Public Member Functions

- **InterruptedIOException** () throw ()
  Default Constructor.
- **InterruptedIOException** (const **lang::Exception** &ex) throw ()
  Copy Constructor.
- **InterruptedIOException** (const **InterruptedIOException** &ex) throw ()
  Copy Constructor.
- **InterruptedIOException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **InterruptedIOException** (const std::exception *__cause__) throw ()
  Constructor.
- **InterruptedIOException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor.
- virtual **InterruptedIOException** * **clone** () const
  Clones this exception.
- virtual ~**InterruptedIOException** () throw ()

6.280.1 Constructor & Destructor Documentation

6.280.1.1 decaf::io::InterruptedIOException::InterruptedIOException () throw () [inline]

Default Constructor.

6.280.1.2 decaf::io::InterruptedIOException::InterruptedIOException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.280.1.3 decaf::io::InterruptedIOException::InterruptedIOException (const InterruptedIOException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.280.1.4 decaf::io::InterruptedIOException::InterruptedIOException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.280.1.5 decaf::io::InterruptedIOException::InterruptedIOException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.280.1.6 decaf::io::InterruptedIOException::InterruptedIOException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.280.1.7 virtual decaf::io::InterruptedIOException::~InterruptedIOException () throw () [inline, virtual]

6.280.2 Member Function Documentation

6.280.2.1 virtual InterruptedIOException* decaf::io::InterruptedIOException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new exception that is a copy of this one.

Reimplemented from **decaf::io::IOException** (p. 1479).

Reimplemented in **decaf::net::SocketTimeoutException** (p. 2397).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**InterruptedIOException.h**

6.281 cms::InvalidClientIdException Class Reference

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

`#include <src/main/cms/InvalidClientIdException.h>`Inheritance diagram for cms::InvalidClientIdException:

Public Member Functions

- **InvalidClientIdException** () throw ()
- **InvalidClientIdException** (const **InvalidClientIdException** &ex) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidClientIdException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidClientIdException** () throw ()

6.281.1 Detailed Description

This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Since:

1.3

6.281.2 Constructor & Destructor Documentation

6.281.2.1 cms::InvalidClientIdException::InvalidClientIdException () throw ()

6.281.2.2 cms::InvalidClientIdException::InvalidClientIdException (const InvalidClientIdException & *ex*) throw ()

6.281.2.3 cms::InvalidClientIdException::InvalidClientIdException (const std::string & *message*, const std::exception * *cause*) throw ()

6.281.2.4 cms::InvalidClientIdException::InvalidClientIdException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.281.2.5 virtual cms::InvalidClientIdException::~InvalidClientIdException () throw () `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidClientIdException.h**

6.282 cms::InvalidDestinationException Class Reference

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

`#include <src/main/cms/InvalidDestinationException.h>`Inheritance diagram for cms::InvalidDestinationException:

Public Member Functions

- **InvalidDestinationException** () throw ()
- **InvalidDestinationException** (const **InvalidDestinationException** &ex) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidDestinationException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidDestinationException** () throw ()

6.282.1 Detailed Description

This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Since:

1.3

6.282.2 Constructor & Destructor Documentation

6.282.2.1 cms::InvalidDestinationException::InvalidDestinationException () throw ()

6.282.2.2 cms::InvalidDestinationException::InvalidDestinationException (const InvalidDestinationException & *ex*) throw ()

6.282.2.3 cms::InvalidDestinationException::InvalidDestinationException (const std::string & *message*, const std::exception * *cause*) throw ()

6.282.2.4 cms::InvalidDestinationException::InvalidDestinationException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.282.2.5 virtual cms::InvalidDestinationException::~InvalidDestinationException () throw () `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidDestinationException.h**

6.283 decaf::security::InvalidKeyException Class Reference

`#include <src/main/decaf/security/InvalidKeyException.h>`Inheritance diagram for decaf::security::InvalidKeyException:

Public Member Functions

- **InvalidKeyException** () throw ()
  Default Constructor.
- **InvalidKeyException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **InvalidKeyException** (const **InvalidKeyException** &ex) throw ()
  Copy Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **InvalidKeyException** (const std::exception *cause) throw ()
  Constructor.
- **InvalidKeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **InvalidKeyException** * **clone** () const
  Clones this exception.
- virtual ~**InvalidKeyException** () throw ()

6.283.1 Constructor & Destructor Documentation

6.283.1.1 decaf::security::InvalidKeyException::InvalidKeyException () throw () [inline]

Default Constructor.

6.283.1.2 decaf::security::InvalidKeyException::InvalidKeyException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.283.1.3 decaf::security::InvalidKeyException::InvalidKeyException (const InvalidKeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.283.1.4 decaf::security::InvalidKeyException::InvalidKeyException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.283.1.5 decaf::security::InvalidKeyException::InvalidKeyException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.283.1.6 decaf::security::InvalidKeyException::InvalidKeyException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.283.1.7 virtual decaf::security::InvalidKeyException::~InvalidKeyException () throw () [inline, virtual]

6.283.2 Member Function Documentation

6.283.2.1 virtual InvalidKeyException* decaf::security::InvalidKeyException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::KeyException** (p. 1574).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**InvalidKeyException.h**

6.284 decaf::nio::InvalidMarkException Class Reference

`#include <src/main/decaf/nio/InvalidMarkException.h>`Inheritance diagram for decaf::nio::InvalidMarkException:

Public Member Functions

- **InvalidMarkException** () throw ()
  Default Constructor.
- **InvalidMarkException** (const **lang::Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **InvalidMarkException** (const **InvalidMarkException** &ex) throw ()
  Copy Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **InvalidMarkException** (const std::exception ***cause**) throw ()
  Constructor.
- **InvalidMarkException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidMarkException** * **clone** () const
  Clones this exception.
- virtual ~**InvalidMarkException** () throw ()

6.284.1 Constructor & Destructor Documentation

6.284.1.1 decaf::nio::InvalidMarkException::InvalidMarkException () throw () [inline]

Default Constructor.

6.284.1.2 decaf::nio::InvalidMarkException::InvalidMarkException (const lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex The Exception whose state data is to be copied into this Exception.

6.284.1.3 decaf::nio::InvalidMarkException::InvalidMarkException (const InvalidMarkException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The Exception whose state data is to be copied into this Exception.

6.284.1.4 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.284.1.5 decaf::nio::InvalidMarkException::InvalidMarkException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.284.1.6 decaf::nio::InvalidMarkException::InvalidMarkException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.284.1.7 virtual decaf::nio::InvalidMarkException::~InvalidMarkException () throw () [inline, virtual]

6.284.2 Member Function Documentation

6.284.2.1 virtual InvalidMarkException* decaf::nio::InvalidMarkException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new Exception instance that is a copy of this Exception.

Reimplemented from **decaf::lang::exceptions::IllegalStateException** (p. 1402).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**InvalidMarkException.h**

6.285 cms::InvalidSelectorException Class Reference

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

`#include <src/main/cms/InvalidSelectorException.h>`Inheritance diagram for cms::InvalidSelectorException:

Public Member Functions

- **InvalidSelectorException** () throw ()
- **InvalidSelectorException** (const **InvalidSelectorException** &ex) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception *cause) throw ()
- **InvalidSelectorException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**InvalidSelectorException** () throw ()

6.285.1 Detailed Description

This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Since:

1.3

6.285.2 Constructor & Destructor Documentation

6.285.2.1 cms::InvalidSelectorException::InvalidSelectorException () throw ()

6.285.2.2 cms::InvalidSelectorException::InvalidSelectorException (const InvalidSelectorException & *ex*) throw ()

6.285.2.3 cms::InvalidSelectorException::InvalidSelectorException (const std::string & *message*, const std::exception * *cause*) throw ()

6.285.2.4 cms::InvalidSelectorException::InvalidSelectorException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.285.2.5 virtual cms::InvalidSelectorException::~InvalidSelectorException () throw () `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**InvalidSelectorException.h**

6.286 decaf::lang::exceptions::InvalidStateException Class Reference

`#include <src/main/decaf/lang/exceptions/InvalidStateException.h>`Inheritance diagram for decaf::lang::exceptions::InvalidStateException:

Public Member Functions

- **InvalidStateException** () throw ()
  Default Constructor.
- **InvalidStateException** (const **Exception** &ex) throw ()
  Conversion Constructor from some other ***Exception*** *(*p. *1268**).*
- **InvalidStateException** (const **InvalidStateException** &ex) throw ()
  Copy Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **InvalidStateException** (const std::exception ***cause**) throw ()
  Constructor.
- **InvalidStateException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **InvalidStateException** * **clone** () const
  Clones this exception.
- virtual ~**InvalidStateException** () throw ()

6.286.1 Constructor & Destructor Documentation

6.286.1.1 decaf::lang::exceptions::InvalidStateException::InvalidStateException () throw () [inline]

Default Constructor.

6.286.1.2 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.286.1.3 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const InvalidStateException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.286.1.4 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.286.1.5 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.286.1.6 decaf::lang::exceptions::InvalidStateException::InvalidStateException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.286.1.7 virtual decaf::lang::exceptions::InvalidStateException::~InvalidStateException () throw () [inline, virtual]

6.286.2 Member Function Documentation

6.286.2.1 virtual InvalidStateException* decaf::lang::exceptions::InvalidStateException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**InvalidStateException.h**

6.287 decaf::io::IOException Class Reference

`#include <src/main/decaf/io/IOException.h>`Inheritance diagram for decaf::io::IOException:

Public Member Functions

- **IOException** () throw ()

  Default Constructor.

- **IOException** (const **lang::Exception** &ex) throw ()

  Copy Constructor.

- **IOException** (const **IOException** &ex) throw ()

  Copy Constructor.

- **IOException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **IOException** (const std::exception *__cause__) throw ()

  Constructor.

- **IOException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor.

- virtual **IOException** * **clone** () const

  Clones this exception.

- virtual ~**IOException** () throw ()

6.287.1 Constructor & Destructor Documentation

6.287.1.1 decaf::io::IOException::IOException () throw () `[inline]`

Default Constructor.

6.287.1.2 decaf::io::IOException::IOException (const lang::Exception & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex the exception to copy

6.287.1.3 decaf::io::IOException::IOException (const IOException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.287.1.4 decaf::io::IOException::IOException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.287.1.5 decaf::io::IOException::IOException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.287.1.6 decaf::io::IOException::IOException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.287.1.7 virtual decaf::io::IOException::~IOException () throw () [inline, virtual]

6.287.2 Member Function Documentation

6.287.2.1 virtual IOException* decaf::io::IOException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new instance of an Exception that is a copy of this instance.

Reimplemented from **decaf::lang::Exception** (p. 1271).

Reimplemented in **decaf::io::EOFException** (p. 1267), **decaf::io::InterruptedIOException** (p. 1464), **decaf::io::UTFDataFormatException** (p. 2685), **decaf::net::BindException** (p. 565), **decaf::net::ConnectException** (p. 940), **decaf::net::HttpRetryException** (p. 1392), **decaf::net::MalformedURLException** (p. 1688), **decaf::net::NoRouteToHostException** (p. 1941), **decaf::net::PortUnreachableException** (p. 2039), **decaf::net::ProtocolException** (p. 2152), **decaf::net::SocketException** (p. 2380), **decaf::net::SocketTimeoutException** (p. 2397), **decaf::net::UnknownHostException** (p. 2631), and **decaf::net::UnknownServiceException** (p. 2634).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**IOException.h**

6.288 activemq::transport::IOTransport Class Reference

Implementation of the **Transport** (p. 2608) interface that performs marshaling of **commands** (p. 59) to IO streams.

`#include <src/main/activemq/transport/IOTransport.h>`Inheritance diagram for activemq::transport::IOTransport:

Public Member Functions

- **IOTransport** ()

  Default Constructor.

- **IOTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)

  *Create an instance of this **Transport** (p. 2608) and assign its WireFormat instance at creation time.*

- virtual ~**IOTransport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Not supported by this class - throws an exception.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Not supported by this class - throws an exception.

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)

  Sets the WireFormat instance to use.

- virtual void **setTransportListener** (**TransportListener** *listener)

  *Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*

- virtual **TransportListener** * **getTransportListener** () const

  *Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*

- virtual void **setInputStream** (**decaf::io::DataInputStream** *is)

  *Sets the input stream for in-coming **commands** (p. 59).*

- virtual void **setOutputStream** (**decaf::io::DataOutputStream** *os)

  *Sets the output stream for out-going **commands** (p. 59).*

- virtual void **start** () throw (cms::CMSException)

*Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59).*

- virtual void **close** () throw (cms::CMSException)

  Stops the polling thread and closes the streams.

- virtual void **run** ()

  Runs the polling thread.

- virtual **Transport** * **narrow** (const std::type_info &typeId)

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED) throw (decaf::io::IOException)

  reconnect to another location

6.288.1 Detailed Description

Implementation of the **Transport** (p. 2608) interface that performs marshaling of **commands** (p. 59) to IO streams. This class does not implement the request method, it only handles oneway messages. A thread polls on the input stream for in-coming **commands** (p. 59). When a command is received, the command listener is notified. The polling thread is not started until the start method is called. The close method will close the associated streams. Close can be called explicitly by the user, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

6.288.2 Constructor & Destructor Documentation

6.288.2.1 activemq::transport::IOTransport::IOTransport ()

Default Constructor.

6.288.2.2 activemq::transport::IOTransport::IOTransport (const Pointer< wireformat::WireFormat > & *wireFormat*)

Create an instance of this **Transport** (p. 2608) and assign its WireFormat instance at creation time.

Parameters:

wireFormat Data encoder / decoder to use when reading and writing.

6.288.2.3 virtual activemq::transport::IOTransport::~IOTransport () [virtual]

6.288.3 Member Function Documentation

6.288.3.1 virtual void activemq::transport::IOTransport::close () throw (cms::CMSException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

CMSException if errors occur.

Implements **cms::Closeable** (p. 838).

6.288.3.2 virtual std::string activemq::transport::IOTransport::getRemoteAddress () const [inline, virtual]

Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2609).

6.288.3.3 virtual TransportListener* activemq::transport::IOTransport::getTransportListener () const [inline, virtual]

Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Returns:

The listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2609).

6.288.3.4 virtual bool activemq::transport::IOTransport::isClosed () const [inline, virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Implements **activemq::transport::Transport** (p. 2609).

6.288.3.5 virtual bool activemq::transport::IOTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2610).

6.288.3.6 virtual bool activemq::transport::IOTransport::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2610).

6.288.3.7 virtual Transport* activemq::transport::IOTransport::narrow (const std::type_info & *typeId*) [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2610).

6.288.3.8 virtual void activemq::transport::IOTransport::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2610).

6.288.3.9 virtual void activemq::transport::IOTransport::reconnect (const decaf::net::URI &uri *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

6.288.3.10 virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command to be sent.

timeout the time to wait for a response.

Returns:

the response to the command sent.

Exceptions:

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2611).

6.288.3.11 virtual Pointer<Response> activemq::transport::IOTransport::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command to be sent.

Returns:

the response to the command sent.

Exceptions:

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2612).

6.288.3.12 virtual void activemq::transport::IOTransport::run () [virtual]

Runs the polling thread.

Implements **decaf::lang::Runnable** (p. 2256).

6.288.3.13 virtual void activemq::transport::IOTransport::setInputStream (decaf::io::DataInputStream * *is*) [inline, virtual]

Sets the input stream for in-coming **commands** (p. 59).

Parameters:

is The input stream.

6.288.3.14 virtual void activemq::transport::IOTransport::setOutputStream (decaf::io::DataOutputStream * *os*) [inline, virtual]

Sets the output stream for out-going **commands** (p. 59).

Parameters:

os The output stream.

6.288.3.15 virtual void activemq::transport::IOTransport::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Parameters:

listener the listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2612).

6.288.3.16 virtual void activemq::transport::IOTransport::setWireFormat (const Pointer< wireformat::WireFormat > & *wireFormat*) [inline, virtual]

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode **commands** (p. 59).

Implements **activemq::transport::Transport** (p. 2612).

6.288.3.17 virtual void activemq::transport::IOTransport::start () throw (cms::CMSException) [virtual]

Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this **transport** (p. 67) has already been closed.

Implements **cms::Startable** (p. 2413).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**IOTransport.h**

6.289 decaf::lang::Iterable< E > Class Template Reference

Implementing this interface allows an object to be cast to an **Iterable** (p. 1487) type for generic collections API calls.

`#include <src/main/decaf/lang/Iterable.h>`Inheritance diagram for decaf::lang::Iterable< E >:

Public Member Functions

- virtual ~**Iterable** ()
- virtual **decaf::util::Iterator**< E > * **iterator** ()=0
- virtual **decaf::util::Iterator**< E > * **iterator** () const =0

6.289.1 Detailed Description

template<typename E> class decaf::lang::Iterable< E >

Implementing this interface allows an object to be cast to an **Iterable** (p. 1487) type for generic collections API calls.

6.289.2 Constructor & Destructor Documentation

6.289.2.1 template<typename E> virtual decaf::lang::Iterable< E >::~Iterable () [inline, virtual]

6.289.3 Member Function Documentation

6.289.3.1 template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator () const [pure virtual]

Implemented in **decaf::util::StlList**< **E** > (p. 2425), **decaf::util::StlSet**< **E** > (p. 2450), **decaf::util::StlList**< **CompositeTask** * > (p. 2425), **decaf::util::StlList**< **URI** > (p. 2425), **decaf::util::StlList**< **Pointer**< **DestinationInfo** > > (p. 2425), **decaf::util::StlList**< **PrimitiveValueNode** > (p. 2425), **decaf::util::StlList**< **Pointer**< **Command** > > (p. 2425), **decaf::util::StlList**< **Pointer**< **BackupTransport** > > (p. 2425), **decaf::util::StlSet**< **Pointer**< **Synchronization** > > (p. 2450), and **decaf::util::StlSet**< **ActiveMQSession** * > (p. 2450).

6.289.3.2 template<typename E> virtual decaf::util::Iterator<E>* decaf::lang::Iterable< E >::iterator () [pure virtual]

Returns:

an iterator over a set of elements of type T.

Implemented in **decaf::util::StlList**< **E** > (p. 2425), **decaf::util::StlSet**< **E** > (p. 2451), **decaf::util::StlList**< **CompositeTask** * > (p. 2425), **decaf::util::StlList**< **URI** > (p. 2425),

decaf::util::StlList< Pointer< DestinationInfo > > (p. 2425), **decaf::util::StlList< PrimitiveValueNode >** (p. 2425), **decaf::util::StlList< Pointer< Command > >** (p. 2425), **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2425), **decaf::util::StlSet< Pointer< Synchronization > >** (p. 2451), and **decaf::util::StlSet< ActiveMQSession * >** (p. 2451).

Referenced by decaf::util::AbstractCollection< Pointer< BackupTransport > >::clear(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::contains(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::copy(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::operator=(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::remove(), decaf::util::AbstractSet< ActiveMQSession * >::removeAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::removeAll(), decaf::util::AbstractCollection< Pointer< BackupTransport > >::retainAll(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::toArray().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Iterable.h**

6.290 decaf::util::Iterator< T > Class Template Reference

Defines an object that can be used to iterate over the elements of a collection.

`#include <src/main/decaf/util/Iterator.h>`Inheritance diagram for decaf::util::Iterator< T >:

Public Member Functions

- virtual ~**Iterator** ()
- virtual T **next** ()=0 throw (lang::exceptions::NoSuchElementException)

  Returns the next element in the iteration.

- virtual bool **hasNext** () const =0

  Returns true if the iteration has more elements.

- virtual void **remove** ()=0 throw (lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException)

  Removes from the underlying collection the last element returned by the iterator (optional operation).

6.290.1 Detailed Description

template<typename T> class decaf::util::Iterator< T >

Defines an object that can be used to iterate over the elements of a collection. The iterator provides a way to access and remove elements with well defined semantics.

6.290.2 Constructor & Destructor Documentation

6.290.2.1 template<typename T> virtual decaf::util::Iterator< T >::~Iterator () [inline, virtual]

6.290.3 Member Function Documentation

6.290.3.1 template<typename T> virtual bool decaf::util::Iterator< T >::hasNext () const [pure virtual]

Returns true if the iteration has more elements. Returns false if the next call to next would result in an NoSuchElementException to be thrown.

6.290.3.2 template<typename T> virtual T decaf::util::Iterator< T >::next () throw (lang::exceptions::NoSuchElementException) [pure virtual]

Returns the next element in the iteration. Calling this method repeatedly until the **hasNext()** (p. 1489) method returns false will return each element in the underlying collection exactly once.

Returns:

next element in the iteration of elements

Exceptions:

NoSuchElementException - iteration has no more elements.

6.290.3.3 template<typename T> virtual void decaf::util::Iterator< T >::remove () throw (lang::exceptions::IllegalStateException, lang::exceptions::UnsupportedOperationException) [pure virtual]

Removes from the underlying collection the last element returned by the iterator (optional operation). This method can be called only once per call to next. The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

Exceptions:

UnsupportedOperationException - if the remove operation is not supported by this **Iterator** (p. 1489).

IllegalStateException - if the next method has not yet been called, or the remove method has already been called after the last call to the next method.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Iterator.h**

6.291 activemq::commands::JournalQueueAck Class Reference

`#include <src/main/activemq/commands/JournalQueueAck.h>`Inheritance diagram for activemq::commands::JournalQueueAck:

Public Member Functions

- **JournalQueueAck** ()
- virtual ~**JournalQueueAck** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalQueueAck** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **MessageAck** > & **getMessageAck** () const
- virtual **Pointer**< **MessageAck** > & **getMessageAck** ()
- virtual void **setMessageAck** (const **Pointer**< **MessageAck** > &**messageAck**)

Static Public Attributes

- static const unsigned char **ID_JOURNALQUEUEACK** = 52

Protected Member Functions

- **JournalQueueAck** (const **JournalQueueAck** &)
- **JournalQueueAck** & **operator**= (const **JournalQueueAck** &)

Protected Attributes

- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **MessageAck** > **messageAck**

6.291.1 Constructor & Destructor Documentation

6.291.1.1 activemq::commands::JournalQueueAck::JournalQueueAck (const JournalQueueAck &) [inline, protected]

6.291.1.2 activemq::commands::JournalQueueAck::JournalQueueAck ()

6.291.1.3 virtual activemq::commands::JournalQueueAck::~JournalQueueAck () [virtual]

6.291.2 Member Function Documentation

6.291.2.1 virtual JournalQueueAck* activemq::commands::JournalQueueAck::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.291.2.2 virtual void activemq::commands::JournalQueueAck::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.291.2.3 virtual bool activemq::commands::JournalQueueAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.291.2.4 virtual unsigned char activemq::commands::JournalQueueAck::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.291.2.5 virtual Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () [virtual]

6.291.2.6 virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalQueueAck::getDestination () const [virtual]

6.291.2.7 virtual Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () [virtual]

6.291.2.8 virtual const Pointer<MessageAck>& activemq::commands::JournalQueueAck::getMessageAck () const [virtual]

6.291.2.9 JournalQueueAck& activemq::commands::JournalQueueAck::operator= (const JournalQueueAck &) [inline, protected]

6.291.2.10 virtual void activemq::commands::JournalQueueAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.291.2.11 virtual void activemq::commands::JournalQueueAck::setMessageAck (const Pointer< MessageAck > & *messageAck*) [virtual]

6.291.2.12 virtual std::string activemq::commands::JournalQueueAck::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.291.3 Field Documentation

6.291.3.1 Pointer<ActiveMQDestination> activemq::commands::JournalQueueAck::destination [protected]

6.291.3.2 const unsigned char activemq::commands::JournalQueueAck::ID_JOURNALQUEUEACK = 52 [static]

6.291.3.3 Pointer<MessageAck> activemq::commands::JournalQueueAck::messageAck [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalQueueAck.h**

6.292 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMa Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1495).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual ~**JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.292.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1495). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.292.2 Constructor & Destructor Documentation

6.292.2.1 activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::JournalQueueA () [inline]

6.292.2.2 virtual activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::~JournalQueu () [inline, virtual]

6.292.3 Member Function Documentation

6.292.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.292.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.292.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.292.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.292.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.292.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.292.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalQueueAckMarshaller.h**

6.293 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMa Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1499).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalQueueAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual ~**JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.293.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1499). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.293.2 Constructor & Destructor Documentation

6.293.2.1 activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::JournalQueueA () [inline]

6.293.2.2 virtual activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::~JournalQueu () [inline, virtual]

6.293.3 Member Function Documentation

6.293.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.293.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.293.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.293.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.293.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.293.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.293.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalQueueAckMarshaller.h**

6.294 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMa Class Reference

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1503).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalQueueAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller:

Public Member Functions

- **JournalQueueAckMarshaller** ()
- virtual ~**JournalQueueAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.294.1 Detailed Description

Marshaling code for Open Wire Format for **JournalQueueAckMarshaller** (p. 1503). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.294.2 Constructor & Destructor Documentation

6.294.2.1 activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::JournalQueueA () [inline]

6.294.2.2 virtual activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::~JournalQueu () [inline, virtual]

6.294.3 Member Function Documentation

6.294.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.294.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.294.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.294.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.294.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.294.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.294.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalQueueAckMarshaller.h**

6.295 activemq::commands::JournalTopicAck Class Reference

`#include <src/main/activemq/commands/JournalTopicAck.h>`Inheritance diagram for activemq::commands::JournalTopicAck:

Public Member Functions

- **JournalTopicAck** ()
- virtual ~**JournalTopicAck** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalTopicAck** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual long long **getMessageSequenceId** () const
- virtual void **setMessageSequenceId** (long long **messageSequenceId**)
- virtual const std::string & **getSubscritionName** () const
- virtual std::string & **getSubscritionName** ()
- virtual void **setSubscritionName** (const std::string &**subscritionName**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &**clientId**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)

Static Public Attributes

- static const unsigned char **ID_JOURNALTOPICACK** = 50

Protected Member Functions

- **JournalTopicAck** (const **JournalTopicAck** &)
- **JournalTopicAck** & **operator=** (const **JournalTopicAck** &)

Protected Attributes

- **Pointer< ActiveMQDestination > destination**
- **Pointer< MessageId > messageId**
- long long **messageSequenceId**
- std::string **subscritionName**
- std::string **clientId**
- **Pointer< TransactionId > transactionId**

6.295.1 Constructor & Destructor Documentation

6.295.1.1 activemq::commands::JournalTopicAck::JournalTopicAck (const JournalTopicAck &) [inline, protected]

6.295.1.2 activemq::commands::JournalTopicAck::JournalTopicAck ()

6.295.1.3 virtual activemq::commands::JournalTopicAck::~JournalTopicAck () [virtual]

6.295.2 Member Function Documentation

6.295.2.1 virtual JournalTopicAck* activemq::commands::JournalTopicAck::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.295.2.2 virtual void activemq::commands::JournalTopicAck::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.295.2.3 virtual bool activemq::commands::JournalTopicAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.295.2.4 virtual std::string& activemq::commands::JournalTopicAck::getClientId () [virtual]

6.295.2.5 virtual const std::string& activemq::commands::JournalTopicAck::getClientId () const [virtual]

6.295.2.6 virtual unsigned char activemq::commands::JournalTopicAck::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.295.2.7 virtual Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () [virtual]

6.295.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::JournalTopicAck::getDestination () const [virtual]

6.295.2.9 virtual Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () [virtual]

6.295.2.10 virtual const Pointer<MessageId>& activemq::commands::JournalTopicAck::getMessageId () const [virtual]

6.295.2.11 virtual long long activemq::commands::JournalTopicAck::getMessageSequenceId () const [virtual]

6.295.2.12 virtual std::string& activemq::commands::JournalTopicAck::getSubscritionName () [virtual]

6.295.2.13 virtual const std::string& activemq::commands::JournalTopicAck::getSubscritionName () const [virtual]

6.295.2.14 virtual Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () [virtual]

6.295.2.15 virtual const Pointer<TransactionId>& activemq::commands::JournalTopicAck::getTransactionId () const [virtual]

6.295.2.16 JournalTopicAck& activemq::commands::JournalTopicAck::operator= (const JournalTopicAck &) [inline, protected]

6.295.2.17 virtual void activemq::commands::JournalTopicAck::setClientId (const std::string & *clientId*) [virtual]

6.295.2.18 virtual void activemq::commands::JournalTopicAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.295.2.19 virtual void activemq::commands::JournalTopicAck::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.295.2.20 virtual void activemq::commands::JournalTopicAck::setMessageSequenceId (long long *messageSequenceId*) [virtual]

6.295.2.21 virtual void activemq::commands::JournalTopicAck::setSubscritionName (const std::string & *subscritionName*) [virtual]

6.295.2.22 virtual void activemq::commands::JournalTopicAck::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.295.2.23 virtual std::string activemq::commands::JournalTopicAck::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.295.3 Field Documentation

6.295.3.1 std::string activemq::commands::JournalTopicAck::clientId [protected]

6.295.3.2 Pointer<ActiveMQDestination> activemq::commands::JournalTopicAck::destination [protected]

6.295.3.3 const unsigned char activemq::commands::JournalTopicAck::ID_JOURNALTOPICACK = 50 [static]

6.295.3.4 Pointer<MessageId> activemq::commands::JournalTopicAck::messageId [protected]

6.295.3.5 long long activemq::commands::JournalTopicAck::messageSequenceId [protected]

6.295.3.6 std::string activemq::commands::JournalTopicAck::subscritionName [protected]

6.295.3.7 Pointer<TransactionId> activemq::commands::JournalTopicAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTopicAck.h**

6.296 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1512).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTopicAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual ~**JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.296.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1512). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.296.2 Constructor & Destructor Documentation

6.296.2.1 activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::JournalTopicAc () [inline]

6.296.2.2 virtual activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::~JournalTopicA () [inline, virtual]

6.296.3 Member Function Documentation

6.296.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.296.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.296.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.296.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.296.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.296.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.296.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTopicAckMarshaller.h**

6.297 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1516).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual ~**JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.297.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1516). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.297.2 Constructor & Destructor Documentation

6.297.2.1 activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::JournalTopicAc () [inline]

6.297.2.2 virtual activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::~JournalTopicA () [inline, virtual]

6.297.3 Member Function Documentation

6.297.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.297.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.297.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.297.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.297.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.297.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.297.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTopicAckMarshaller.h**

6.298 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMa Class Reference

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1520).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller:

Public Member Functions

- **JournalTopicAckMarshaller** ()
- virtual ~**JournalTopicAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.298.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTopicAckMarshaller** (p. 1520). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.298.2 Constructor & Destructor Documentation

6.298.2.1 activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::JournalTopicAc () [inline]

6.298.2.2 virtual activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::~JournalTopicA () [inline, virtual]

6.298.3 Member Function Documentation

6.298.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.298.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.298.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.298.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.298.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.298.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.298.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTopicAckMarshaller.h**

6.299 activemq::commands::JournalTrace Class Reference

`#include <src/main/activemq/commands/JournalTrace.h>`Inheritance diagram for activemq::commands::JournalTrace:

Public Member Functions

- **JournalTrace** ()
- virtual ~**JournalTrace** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalTrace** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getMessage** () const
- virtual std::string & **getMessage** ()
- virtual void **setMessage** (const std::string &**message**)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRACE** = 53

Protected Member Functions

- **JournalTrace** (const **JournalTrace** &)
- **JournalTrace** & **operator**= (const **JournalTrace** &)

Protected Attributes

- std::string **message**

6.299.1 Constructor & Destructor Documentation

6.299.1.1 activemq::commands::JournalTrace::JournalTrace (const JournalTrace &) [inline, protected]

6.299.1.2 activemq::commands::JournalTrace::JournalTrace ()

6.299.1.3 virtual activemq::commands::JournalTrace::~JournalTrace () [virtual]

6.299.2 Member Function Documentation

6.299.2.1 virtual JournalTrace* activemq::commands::JournalTrace::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.299.2.2 virtual void activemq::commands::JournalTrace::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.299.2.3 virtual bool activemq::commands::JournalTrace::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.299.2.4 virtual unsigned char activemq::commands::JournalTrace::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.299.2.5 virtual std::string& activemq::commands::JournalTrace::getMessage () [virtual]

6.299.2.6 virtual const std::string& activemq::commands::JournalTrace::getMessage () const [virtual]

6.299.2.7 JournalTrace& activemq::commands::JournalTrace::operator= (const JournalTrace &) [inline, protected]

6.299.2.8 virtual void activemq::commands::JournalTrace::setMessage (const std::string & *message*) [virtual]

6.299.2.9 virtual std::string activemq::commands::JournalTrace::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.299.3 Field Documentation

6.299.3.1 const unsigned char activemq::commands::JournalTrace::ID_JOURNALTRACE = 53 [static]

6.299.3.2 std::string activemq::commands::JournalTrace::message [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTrace.h**

6.300 activemq::wireformat::openwire::marshal::v2::JournalTraceMarsha Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1527).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTraceMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.300.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1527). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.300.2 Constructor & Destructor Documentation

6.300.2.1 activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::JournalTraceMarsh () [inline]

6.300.2.2 virtual activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::~JournalTraceMars () [inline, virtual]

6.300.3 Member Function Documentation

6.300.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.300.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.300.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.300.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.300.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.300.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.300.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTraceMarshaller.h**

6.301 activemq::wireformat::openwire::marshal::v3::JournalTraceMarsha Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1531).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTraceMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.301.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1531). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.301.2 Constructor & Destructor Documentation

6.301.2.1 activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::JournalTraceMarsh () [inline]

6.301.2.2 virtual activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::~JournalTraceMars () [inline, virtual]

6.301.3 Member Function Documentation

6.301.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.301.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.301.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.301.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.301.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.301.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.301.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTraceMarshaller.h**

6.302 activemq::wireformat::openwire::marshal::v1::JournalTraceMarsha Class Reference

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1535).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTraceMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller:

Public Member Functions

- **JournalTraceMarshaller** ()
- virtual ~**JournalTraceMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.302.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTraceMarshaller** (p. 1535). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.302.2 Constructor & Destructor Documentation

6.302.2.1 activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::JournalTraceMarsh () [inline]

6.302.2.2 virtual activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::~JournalTraceMars () [inline, virtual]

6.302.3 Member Function Documentation

6.302.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.302.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.302.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.302.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.302.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.302.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.302.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTraceMarshaller.h**

6.303 activemq::commands::JournalTransaction Class Reference

`#include <src/main/activemq/commands/JournalTransaction.h>`Inheritance diagram for activemq::commands::JournalTransaction:

Public Member Functions

- **JournalTransaction** ()
- virtual ~**JournalTransaction** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **JournalTransaction** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char **type**)
- virtual bool **getWasPrepared** () const
- virtual void **setWasPrepared** (bool **wasPrepared**)

Static Public Attributes

- static const unsigned char **ID_JOURNALTRANSACTION** = 54

Protected Member Functions

- **JournalTransaction** (const **JournalTransaction** &)
- **JournalTransaction** & **operator**= (const **JournalTransaction** &)

Protected Attributes

- **Pointer< TransactionId > transactionId**
- unsigned char **type**
- bool **wasPrepared**

6.303.1 Constructor & Destructor Documentation

6.303.1.1 activemq::commands::JournalTransaction::JournalTransaction (const JournalTransaction &) [inline, protected]

6.303.1.2 activemq::commands::JournalTransaction::JournalTransaction ()

6.303.1.3 virtual activemq::commands::JournalTransaction::~JournalTransaction () [virtual]

6.303.2 Member Function Documentation

6.303.2.1 virtual JournalTransaction* activemq::commands::JournalTransaction::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.303.2.2 virtual void activemq::commands::JournalTransaction::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.303.2.3 virtual bool activemq::commands::JournalTransaction::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.303.2.4 virtual unsigned char activemq::commands::JournalTransaction::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.303.2.5 virtual Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () [virtual]

6.303.2.6 virtual const Pointer<TransactionId>& activemq::commands::JournalTransaction::getTransactionId () const [virtual]

6.303.2.7 virtual unsigned char activemq::commands::JournalTransaction::getType () const [virtual]

6.303.2.8 virtual bool activemq::commands::JournalTransaction::getWasPrepared () const [virtual]

6.303.2.9 JournalTransaction& activemq::commands::JournalTransaction::operator= (const JournalTransaction &) [inline, protected]

6.303.2.10 virtual void activemq::commands::JournalTransaction::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.303.2.11 virtual void activemq::commands::JournalTransaction::setType (unsigned char *type*) [virtual]

6.303.2.12 virtual void activemq::commands::JournalTransaction::setWasPrepared (bool *wasPrepared*) [virtual]

6.303.2.13 virtual std::string activemq::commands::JournalTransaction::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.303.3 Field Documentation

6.303.3.1 const unsigned char activemq::commands::JournalTransaction::ID_JOURNALTRANSACTION = 54 [static]

6.303.3.2 Pointer<TransactionId> activemq::commands::JournalTransaction::transactionId [protected]

6.303.3.3 unsigned char activemq::commands::JournalTransaction::type [protected]

6.303.3.4 bool activemq::commands::JournalTransaction::wasPrepared [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**JournalTransaction.h**

6.304 activemq::wireformat::openwire::marshal::v2::JournalTransactionM Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1543).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/JournalTransactionMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.304.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1543). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.304.2 Constructor & Destructor Documentation

6.304.2.1 activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::JournalTrans () [inline]

6.304.2.2 virtual activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::~JournalTra () [inline, virtual]

6.304.3 Member Function Documentation

6.304.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.304.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.304.3.3 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.304.3.4 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.304.3.5 virtual int activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.304.3.6 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.304.3.7 virtual void activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**JournalTransactionMarshaller.h**

6.305 activemq::wireformat::openwire::marshal::v3::JournalTransactionM Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1547).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/JournalTransactionMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.305.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1547). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.305.2 Constructor & Destructor Documentation

6.305.2.1 activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::JournalTrans () [inline]

6.305.2.2 virtual activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::~JournalTra () [inline, virtual]

6.305.3 Member Function Documentation

6.305.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.305.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.305.3.3 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.305.3.4 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.305.3.5 virtual int activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.305.3.6 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.305.3.7 virtual void activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**JournalTransactionMarshaller.h**

6.306 activemq::wireformat::openwire::marshal::v1::JournalTransactionM Class Reference

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1551).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller:

Public Member Functions

- **JournalTransactionMarshaller** ()
- virtual ~**JournalTransactionMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.306.1 Detailed Description

Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1551). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.306.2 Constructor & Destructor Documentation

6.306.2.1 activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::JournalTrans () [inline]

6.306.2.2 virtual activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::~JournalTra () [inline, virtual]

6.306.3 Member Function Documentation

6.306.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.306.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.306.3.3 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.306.3.4 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.306.3.5 virtual int activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.306.3.6 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.306.3.7 virtual void activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**JournalTransactionMarshaller.h**

6.307 activemq::commands::KeepAliveInfo Class Reference

`#include <src/main/activemq/commands/KeepAliveInfo.h>`Inheritance diagram for activemq::commands::KeepAliveInfo:

Public Member Functions

- **KeepAliveInfo** ()
- virtual ~**KeepAliveInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **KeepAliveInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual bool **isKeepAliveInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_KEEPALIVEINFO** = 10

Protected Member Functions

- **KeepAliveInfo** (const **KeepAliveInfo** &)
- **KeepAliveInfo** & **operator**= (const **KeepAliveInfo** &)

6.307.1 Constructor & Destructor Documentation

6.307.1.1 activemq::commands::KeepAliveInfo::KeepAliveInfo (const KeepAliveInfo &) [inline, protected]

6.307.1.2 activemq::commands::KeepAliveInfo::KeepAliveInfo ()

6.307.1.3 virtual activemq::commands::KeepAliveInfo::~KeepAliveInfo () [virtual]

6.307.2 Member Function Documentation

6.307.2.1 virtual KeepAliveInfo* activemq::commands::KeepAliveInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.307.2.2 virtual void activemq::commands::KeepAliveInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.307.2.3 virtual bool activemq::commands::KeepAliveInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.307.2.4 virtual unsigned char activemq::commands::KeepAliveInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.307.2.5 virtual bool activemq::commands::KeepAliveInfo::isKeepAliveInfo () const [inline, virtual]

Returns:

an answer of true to the **isKeepAliveInfo()** (p. 1557) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.307.2.6 KeepAliveInfo& activemq::commands::KeepAliveInfo::operator= (const KeepAliveInfo &) [inline, protected]

6.307.2.7 virtual std::string activemq::commands::KeepAliveInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.307.2.8 virtual Pointer<Command> activemq::commands::KeepAliveInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.307.3 Field Documentation

6.307.3.1 const unsigned char activemq::commands::KeepAliveInfo::ID_KEEPALIVEINFO = 10 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**KeepAliveInfo.h**

6.308 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarsh Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1558).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/KeepAliveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual ~**KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.308.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1558). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.308.2 Constructor & Destructor Documentation

6.308.2.1 activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::KeepAliveInfoMar () [inline]

6.308.2.2 virtual activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::~KeepAliveInfoM () [inline, virtual]

6.308.3 Member Function Documentation

6.308.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.308.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.308.3.3 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.308.3.4 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.308.3.5 virtual int activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.308.3.6 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.308.3.7 virtual void activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**KeepAliveInfoMarshaller.h**

6.309 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarsh Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1562).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual ~**KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.309.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1562). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.309.2 Constructor & Destructor Documentation

6.309.2.1 activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::KeepAliveInfoMar () [inline]

6.309.2.2 virtual activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::~KeepAliveInfoM () [inline, virtual]

6.309.3 Member Function Documentation

6.309.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.309.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.309.3.3 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.309.3.4 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.309.3.5 virtual int activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.309.3.6 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.309.3.7 virtual void activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**KeepAliveInfoMarshaller.h**

6.310 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarsh Class Reference

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1566).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/KeepAliveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller:

Public Member Functions

- **KeepAliveInfoMarshaller** ()
- virtual ~**KeepAliveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.310.1 Detailed Description

Marshaling code for Open Wire Format for **KeepAliveInfoMarshaller** (p. 1566). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.310.2 Constructor & Destructor Documentation

6.310.2.1 activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::KeepAliveInfoMar () [inline]

6.310.2.2 virtual activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::~KeepAliveInfoM () [inline, virtual]

6.310.3 Member Function Documentation

6.310.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.310.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.310.3.3 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.310.3.4 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.310.3.5 virtual int activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.310.3.6 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.310.3.7 virtual void activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**KeepAliveInfoMarshaller.h**

6.311 decaf::security::Key Class Reference

The **Key** (p. 1570) interface is the top-level interface for all keys.

`#include <src/main/decaf/security/Key.h>`Inheritance diagram for decaf::security::Key:

Public Member Functions

- virtual ~**Key** ()
- virtual std::string **getAlgorithm** () const =0

  Returns the standard algorithm name for this key.

- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0

  Provides the key in its primary encoding format, or nothing if this key does not support encoding.

- virtual std::string **getFormat** () const =0

  Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding.

6.311.1 Detailed Description

The **Key** (p. 1570) interface is the top-level interface for all keys. It defines the functionality shared by all key objects. All keys have three characteristics:

An Algorithm

This is the key algorithm for that key. The key algorithm is usually an encryption or asymmetric operation algorithm (such as DSA or RSA), which will work with those algorithms and with related algorithms (such as MD5 with RSA, SHA-1 with RSA, Raw DSA, etc.) The name of the algorithm of a key is obtained using the getAlgorithm method.

An Encoded Form

This is an external encoded form for the key used when a standard representation of the key is needed outside the application, as when transmitting the key to some other party. The key is encoded according to a standard format (such as X.509 SubjectPublicKeyInfo or PKCS#8), and is returned using the getEncoded method. Note: The syntax of the ASN.1 type SubjectPublicKeyInfo is defined as follows:

SubjectPublicKeyInfo ::= SEQUENCE {

algorithm AlgorithmIdentifier,

subjectPublicKey BIT STRING }

AlgorithmIdentifier ::= SEQUENCE {

algorithm OBJECT IDENTIFIER,

parameters ANY DEFINED BY algorithm OPTIONAL }

For more information, see RFC 2459: Internet X.509 Public **Key** (p. 1570) Infrastructure Certificate and CRL Profile.

A Format

This is the name of the format of the encoded key. It is returned by the getFormat method.

6.311.2 Constructor & Destructor Documentation

6.311.2.1 virtual decaf::security::Key::~Key () [inline, virtual]

6.311.3 Member Function Documentation

6.311.3.1 virtual std::string decaf::security::Key::getAlgorithm () const [pure virtual]

Returns the standard algorithm name for this key. For example, "DSA" would indicate that this key is a DSA key.

Returns:

the name of the algorithm associated with this key.

6.311.3.2 virtual void decaf::security::Key::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

Provides the key in its primary encoding format, or nothing if this key does not support encoding.

Parameters:

output Receives the encoded key, or nothing if the key does not support encoding.

6.311.3.3 virtual std::string decaf::security::Key::getFormat () const [pure virtual]

Returns the name of the primary encoding format of this key, or an empty string if this key does not support encoding. The primary encoding format is named in terms of the appropriate ASN.1 data format, if an ASN.1 specification for this key exists. For example, the name of the ASN.1 data format for public keys is SubjectPublicKeyInfo, as defined by the X.509 standard; in this case, the returned format is "X.509". Similarly, the name of the ASN.1 data format for private keys is PrivateKeyInfo, as defined by the PKCS #8 standard; in this case, the returned format is "PKCS#8".

Returns:

the primary encoding format of the key.

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Key.h**

6.312 decaf::security::KeyException Class Reference

`#include <src/main/decaf/security/KeyException.h>`Inheritance diagram for decaf::security::KeyException:

Public Member Functions

- **KeyException** () throw ()

  Default Constructor.

- **KeyException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **KeyException** (const **KeyException** &ex) throw ()

  Copy Constructor.

- **KeyException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **KeyException** (const std::exception *cause) throw ()

  Constructor.

- **KeyException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual **KeyException** * **clone** () const

  Clones this exception.

- virtual ~**KeyException** () throw ()

6.312.1 Constructor & Destructor Documentation

6.312.1.1 decaf::security::KeyException::KeyException () throw () [inline]

Default Constructor.

6.312.1.2 decaf::security::KeyException::KeyException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.312.1.3 decaf::security::KeyException::KeyException (const KeyException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.312.1.4 decaf::security::KeyException::KeyException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.312.1.5 decaf::security::KeyException::KeyException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.312.1.6 decaf::security::KeyException::KeyException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.312.1.7 virtual decaf::security::KeyException::~KeyException () throw () [inline, virtual]

6.312.2 Member Function Documentation

6.312.2.1 virtual KeyException* decaf::security::KeyException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1381).

Reimplemented in **decaf::security::InvalidKeyException** (p. 1469).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**KeyException.h**

6.313 activemq::commands::LastPartialCommand Class Reference

`#include <src/main/activemq/commands/LastPartialCommand.h>`Inheritance diagram for activemq::commands::LastPartialCommand:

Public Member Functions

- **LastPartialCommand** ()
- virtual ~**LastPartialCommand** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **LastPartialCommand** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

Static Public Attributes

- static const unsigned char **ID_LASTPARTIALCOMMAND** = 61

Protected Member Functions

- **LastPartialCommand** (const **LastPartialCommand** &)
- **LastPartialCommand** & **operator=** (const **LastPartialCommand** &)

6.313.1 Constructor & Destructor Documentation

6.313.1.1 activemq::commands::LastPartialCommand::LastPartialCommand (const LastPartialCommand &) [inline, protected]

6.313.1.2 activemq::commands::LastPartialCommand::LastPartialCommand ()

6.313.1.3 virtual activemq::commands::LastPartialCommand::~LastPartialCommand () [virtual]

6.313.2 Member Function Documentation

6.313.2.1 virtual LastPartialCommand* activemq::commands::LastPartialCommand::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::PartialCommand** (p. 1996).

6.313.2.2 virtual void activemq::commands::LastPartialCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::PartialCommand** (p. 1996).

6.313.2.3 virtual bool activemq::commands::LastPartialCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::PartialCommand** (p. 1996).

6.313.2.4 virtual unsigned char activemq::commands::LastPartialCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::PartialCommand** (p. 1997).

6.313.2.5 LastPartialCommand& activemq::commands::LastPartialCommand::operator= (const LastPartialCommand &) [inline, protected]

Reimplemented from **activemq::commands::PartialCommand** (p. 1997).

6.313.2.6 virtual std::string activemq::commands::LastPartialCommand::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::PartialCommand** (p. 1997).

6.313.3 Field Documentation

6.313.3.1 const unsigned char activemq::commands::LastPartialCommand::ID_LASTPARTIALCOMMAND = 61 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LastPartialCommand.h**

6.314 activemq::wireformat::openwire::marshal::v3::LastPartialComman Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1578).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/LastPartialCommandMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual ~**LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.314.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1578). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.314.2 Constructor & Destructor Documentation

6.314.2.1 activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::LastPartial () [inline]

6.314.2.2 virtual activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::~LastParti () [inline, virtual]

6.314.3 Member Function Documentation

6.314.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004).

6.314.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004).

6.314.3.3 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2004).

6.314.3.4 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2005).

6.314.3.5 virtual int activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2005).

6.314.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2006).

6.314.3.7 virtual void activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller** (p. 2006).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LastPartialCommandMarshaller.h**

6.315 activemq::wireformat::openwire::marshal::v1::LastPartialComman Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1582).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/LastPartialCommandMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual ~**LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.315.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1582). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.315.2 Constructor & Destructor Documentation

6.315.2.1 activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::LastPartial () [inline]

6.315.2.2 virtual activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::~LastParti () [inline, virtual]

6.315.3 Member Function Documentation

6.315.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2008).

6.315.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2008).

6.315.3.3 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2008).

6.315.3.4 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2009).

6.315.3.5 virtual int activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2009).

6.315.3.6 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2010).

6.315.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller** (p. 2010).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LastPartialCommandMarshaller.h**

6.316 activemq::wireformat::openwire::marshal::v2::LastPartialComman Class Reference

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1586).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/LastPartialCommandMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller:

Public Member Functions

- **LastPartialCommandMarshaller** ()
- virtual ~**LastPartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.316.1 Detailed Description

Marshaling code for Open Wire Format for **LastPartialCommandMarshaller** (p. 1586). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.316.2 Constructor & Destructor Documentation

6.316.2.1 activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::LastPartial () [inline]

6.316.2.2 virtual activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::~LastParti () [inline, virtual]

6.316.3 Member Function Documentation

6.316.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2000).

6.316.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::getDataStruc () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2000).

6.316.3.3 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2000).

6.316.3.4 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2001).

6.316.3.5 virtual int activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2001).

6.316.3.6 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2002).

6.316.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller** (p. 2002).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LastPartialCommandMarshaller.h**

6.317 std::less< decaf::lang::Pointer< T > > Struct Template Reference

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

`#include <src/main/decaf/lang/Pointer.h>`Inheritance diagram for std::less< decaf::lang::Pointer< T > >:

Public Member Functions

- bool **operator()** (const **decaf::lang::Pointer**< T > &left, const **decaf::lang::Pointer**< T > &right) const

6.317.1 Detailed Description

template<typename T> struct std::less< decaf::lang::Pointer< T > >

An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

6.317.2 Member Function Documentation

6.317.2.1 template<typename T > bool std::less< decaf::lang::Pointer< T > >::operator() (const decaf::lang::Pointer< T > & *left*, const decaf::lang::Pointer< T > & *right*) const [inline]

References decaf::lang::Pointer< T, REFCOUNTER >::get().

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.318 decaf::util::List< E > Class Template Reference

An ordered collection (also known as a sequence).

`#include <src/main/decaf/util/List.h>`Inheritance diagram for decaf::util::List< E >:

Public Member Functions

- virtual ~**List** ()
- virtual **ListIterator**< E > * **listIterator** ()=0
- virtual **ListIterator**< E > * **listIterator** () const =0
- virtual **ListIterator**< E > * **listIterator** (std::size_t index)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual std::size_t **indexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)

  Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual size_t **lastIndexOf** (const E &value)=0 throw (decaf::lang::exceptions::NoSuchElementException)

  Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

- virtual E **get** (std::size_t index) const =0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Gets the element contained at position passed.

- virtual E **set** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Replaces the element at the specified position in this list with the specified element.

- virtual void **add** (std::size_t index, const E &element)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

  Inserts the specified element at the specified position in this list.

- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

  Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

- virtual E **remove** (std::size_t index)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

  Removes the element at the specified position in this list.

6.318.1 Detailed Description

template<typename E> class decaf::util::List< E >

An ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list.

Unlike sets, lists typically allow duplicate elements. More formally, lists typically allow pairs of elements e1 and e2 such that e1.equals(e2), and they typically allow multiple null elements if they allow null elements at all. It is not inconceivable that someone might wish to implement a list that prohibits duplicates, by throwing runtime exceptions when the user attempts to insert them, but we expect this usage to be rare.

6.318.2 Constructor & Destructor Documentation

6.318.2.1 template<typename E> virtual decaf::util::List< E >::~List () [inline, virtual]

6.318.3 Member Function Documentation

6.318.3.1 template<typename E> virtual void decaf::util::List< E >::add (std::size_t *index*, const E & *element*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 2421), **decaf::util::StlList< CompositeTask * >** (p. 2421), **decaf::util::StlList< URI >** (p. 2421), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2421), **decaf::util::StlList< PrimitiveValueNode >** (p. 2421), **decaf::util::StlList< Pointer< Command > >** (p. 2421), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2421).

6.318.3.2 template<typename E> virtual bool decaf::util::List< E >::addAll (std::size_t *index*, const Collection< E > & *source*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Inserts all of the elements in the specified collection into this list at the specified position (optional operation). Shifts the element currently at that position (if any) and any subsequent elements to

the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 871) containing elements to be added to this list

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 2422), **decaf::util::StlList< CompositeTask * >** (p. 2422), **decaf::util::StlList< URI >** (p. 2422), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2422), **decaf::util::StlList< PrimitiveValueNode >** (p. 2422), **decaf::util::StlList< Pointer< Command > >** (p. 2422), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2422).

6.318.3.3 template<typename E> virtual E decaf::util::List< E >::get (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Gets the element contained at position passed.

Parameters:

index - position to get

Returns:

value at index

Implemented in **decaf::util::StlList< E >** (p. 2424), **decaf::util::StlList< CompositeTask * >** (p. 2424), **decaf::util::StlList< URI >** (p. 2424), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2424), **decaf::util::StlList< PrimitiveValueNode >** (p. 2424), **decaf::util::StlList< Pointer< Command > >** (p. 2424), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2424).

6.318.3.4 template<typename E> virtual std::size_t decaf::util::List< E >::indexOf (const E & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the lowest index i such that get(i) == value, or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException if value is not in the list

Implemented in **decaf::util::StlList< E >** (p. 2424), **decaf::util::StlList< CompositeTask * >** (p. 2424), **decaf::util::StlList< URI >** (p. 2424), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2424), **decaf::util::StlList< PrimitiveValueNode >** (p. 2424), **decaf::util::StlList< Pointer< Command > >** (p. 2424), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2424).

6.318.3.5 template<typename E> virtual size_t decaf::util::List< E >::lastIndexOf (const E & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element. More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the last occurrence of the specified element in this list.

Exceptions:

NoSuchElementException if value is not in the list

Implemented in **decaf::util::StlList< E >** (p. 2425), **decaf::util::StlList< CompositeTask * >** (p. 2425), **decaf::util::StlList< URI >** (p. 2425), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2425), **decaf::util::StlList< PrimitiveValueNode >** (p. 2425), **decaf::util::StlList< Pointer< Command > >** (p. 2425), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2425).

6.318.3.6 template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Implemented in **decaf::util::StlList< E >** (p. 2425), **decaf::util::StlList< CompositeTask * >** (p. 2425), **decaf::util::StlList< URI >** (p. 2425), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2425), **decaf::util::StlList< PrimitiveValueNode >** (p. 2425), **decaf::util::StlList< Pointer< Command > >** (p. 2425), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2425).

6.318.3.7 template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator (std::size_t *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Parameters:

index index of first element to be returned from the list iterator (by a call to the next method).

Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException if the index is out of range (index < 0 || index > **size()** (p. 878))

Implemented in **decaf::util::StlList< E >** (p. 2426), **decaf::util::StlList< CompositeTask * >** (p. 2426), **decaf::util::StlList< URI >** (p. 2426), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2426), **decaf::util::StlList< PrimitiveValueNode >** (p. 2426), **decaf::util::StlList< Pointer< Command > >** (p. 2426), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2426).

6.318.3.8 template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () const [pure virtual]

Implemented in **decaf::util::StlList< E >** (p. 2426), **decaf::util::StlList< CompositeTask * >** (p. 2426), **decaf::util::StlList< URI >** (p. 2426), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2426), **decaf::util::StlList< PrimitiveValueNode >** (p. 2426), **decaf::util::StlList< Pointer< Command > >** (p. 2426), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2426).

6.318.3.9 template<typename E> virtual ListIterator<E>* decaf::util::List< E >::listIterator () [pure virtual]

Returns:

a list iterator over the elements in this list (in proper sequence).

Implemented in **decaf::util::StlList< E >** (p. 2426), **decaf::util::StlList< CompositeTask * >** (p. 2426), **decaf::util::StlList< URI >** (p. 2426), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2426), **decaf::util::StlList< PrimitiveValueNode >** (p. 2426), **decaf::util::StlList< Pointer< Command > >** (p. 2426), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2426).

6.318.3.10 template<typename E> virtual E decaf::util::List< E >::remove (std::size_t *index*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters:

index - the index of the element to be removed

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implemented in **decaf::util::StlList< E >** (p. 2426), **decaf::util::StlList< CompositeTask * >** (p. 2426), **decaf::util::StlList< URI >** (p. 2426), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2426), **decaf::util::StlList< PrimitiveValueNode >** (p. 2426), **decaf::util::StlList< Pointer< Command > >** (p. 2426), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2426).

6.318.3.11 template<typename E> virtual E decaf::util::List< E >::set (std::size_t *index*, const E & *element*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters:

index - index of the element to replace

element - element to be stored at the specified position

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

Implemented in **decaf::util::StlList< E >** (p. 2427), **decaf::util::StlList< CompositeTask * >** (p. 2427), **decaf::util::StlList< URI >** (p. 2427), **decaf::util::StlList< Pointer< DestinationInfo > >** (p. 2427), **decaf::util::StlList< PrimitiveValueNode >** (p. 2427), **decaf::util::StlList< Pointer< Command > >** (p. 2427), and **decaf::util::StlList< Pointer< BackupTransport > >** (p. 2427).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**List.h**

6.319 decaf::util::ListIterator< E > Class Template Reference

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

`#include <src/main/decaf/util/ListIterator.h>`Inheritance diagram for decaf::util::ListIterator< E >:

Public Member Functions

- virtual ~**ListIterator** ()
- virtual void **add** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException)

  Inserts the specified element into the list (optional operation).

- virtual void **set** (const E &e)=0 throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException)

  Replaces the last element returned by next or previous with the specified element (optional operation).

- virtual bool **hasPrevious** () const =0

  Returns true if this list iterator has more elements when traversing the list in the reverse direction.

- virtual E **previous** ()=0

  Returns the previous element in the list.

- virtual int **nextIndex** () const =0

  Returns the index of the element that would be returned by a subsequent call to next.

- virtual int **previousIndex** () const =0

  Returns the index of the element that would be returned by a subsequent call to previous.

6.319.1 Detailed Description

template<typename E> class decaf::util::ListIterator< E >

An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list. Note that the **remove()** (p. 1490) and set(Object) methods are not defined in terms of the cursor position; they are defined to operate on the last element returned by a call to **next()** (p. 1489) or **previous()** (p. 1599).

6.319.2 Constructor & Destructor Documentation

6.319.2.1 template<typename E > virtual decaf::util::ListIterator< E >::~ListIterator () [inline, virtual]

6.319.3 Member Function Documentation

6.319.3.1 template<typename E > virtual void decaf::util::ListIterator< E >::add (const E & *e*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException) [pure virtual]

Inserts the specified element into the list (optional operation). The element is inserted immediately before the next element that would be returned by next, if any, and after the next element that would be returned by previous, if any. (If the list contains no elements, the new element becomes the sole element on the list.) The new element is inserted before the implicit cursor: a subsequent call to next would be unaffected, and a subsequent call to previous would return the new element. (This call increases by one the value that would be returned by a call to nextIndex or previousIndex.)

Parameters:

e - the element to insert.

Exceptions:

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

6.319.3.2 template<typename E > virtual bool decaf::util::ListIterator< E >::hasPrevious () const [pure virtual]

Returns true if this list iterator has more elements when traversing the list in the reverse direction. (In other words, returns true if previous would return an element rather than throwing an exception.)

Returns:

true if the list iterator has more elements when traversing the list in the reverse direction.

6.319.3.3 template<typename E > virtual int decaf::util::ListIterator< E >::nextIndex () const [pure virtual]

Returns the index of the element that would be returned by a subsequent call to next. (Returns list size if the list iterator is at the end of the list.)

Returns:

the index of the element that would be returned by a subsequent call to next, or list size if list iterator is at end of list.

6.319.3.4 template<typename E > virtual E decaf::util::ListIterator< E >::previous () [pure virtual]

Returns the previous element in the list. This method may be called repeatedly to iterate through the list backwards, or intermixed with calls to next to go back and forth. (Note that alternating calls to next and previous will return the same element repeatedly.)

Returns:

the previous element in the list.

Exceptions:

NoSuchElementException - if the iteration has no previous element.

6.319.3.5 template<typename E > virtual int decaf::util::ListIterator< E >::previousIndex () const [pure virtual]

Returns the index of the element that would be returned by a subsequent call to previous. (Returns -1 if the list iterator is at the beginning of the list.)

Returns:

the index of the element that would be returned by a subsequent call to previous, or -1 if list iterator is at beginning of list.

6.319.3.6 template<typename E > virtual void decaf::util::ListIterator< E >::set (const E & *e*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::IllegalStateException) [pure virtual]

Replaces the last element returned by next or previous with the specified element (optional operation). This call can be made only if neither **ListIterator.remove** (p. 1490) nor **ListIterator.add** (p. 1598) have been called after the last call to next or previous.

Parameters:

e - the element with which to replace the last element returned by next or previous.

Exceptions:

UnsupportedOperationException - if the add method is not supported by this list iterator.

IllegalArgumentException - if some aspect of this element prevents it from being added to this list.

IllegalStateException - if neither next nor previous have been called, or remove or add have been called after the last call to next or previous.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**ListIterator.h**

6.320 activemq::commands::LocalTransactionId Class Reference

`#include <src/main/activemq/commands/LocalTransactionId.h>`Inheritance diagram for activemq::commands::LocalTransactionId:

Public Types

- typedef **decaf::lang::PointerComparator< LocalTransactionId > COMPARATOR**

Public Member Functions

- **LocalTransactionId** ()
- **LocalTransactionId** (const **LocalTransactionId** &other)
- virtual ~**LocalTransactionId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **LocalTransactionId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual const **Pointer< ConnectionId >** & **getConnectionId** () const
- virtual **Pointer< ConnectionId >** & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer< ConnectionId >** &**connectionId**)
- virtual int **compareTo** (const **LocalTransactionId** &**value**) const
- virtual bool **equals** (const **LocalTransactionId** &**value**) const
- virtual bool **operator==** (const **LocalTransactionId** &**value**) const
- virtual bool **operator<** (const **LocalTransactionId** &**value**) const
- **LocalTransactionId** & **operator=** (const **LocalTransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_LOCALTRANSACTIONID** = 111

Protected Attributes

- long long **value**
- **Pointer< ConnectionId > connectionId**

6.320.1 Member Typedef Documentation

6.320.1.1 typedef decaf::lang::PointerComparator<LocalTransactionId> activemq::commands::LocalTransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2573).

6.320.2 Constructor & Destructor Documentation

6.320.2.1 activemq::commands::LocalTransactionId::LocalTransactionId ()

6.320.2.2 activemq::commands::LocalTransactionId::LocalTransactionId (const LocalTransactionId & *other*)

6.320.2.3 virtual activemq::commands::LocalTransactionId::~LocalTransactionId () [virtual]

6.320.3 Member Function Documentation

6.320.3.1 virtual LocalTransactionId* activemq::commands::LocalTransactionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.320.3.2 virtual int activemq::commands::LocalTransactionId::compareTo (const LocalTransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.320.3.3 virtual void activemq::commands::LocalTransactionId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.320.3.4 virtual bool activemq::commands::LocalTransactionId::equals (const LocalTransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.320.3.5 virtual bool activemq::commands::LocalTransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.3.6 virtual Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () [virtual]

6.320.3.7 virtual const Pointer<ConnectionId>& activemq::commands::LocalTransactionId::getConnectionId () const [virtual]

6.320.3.8 virtual unsigned char activemq::commands::LocalTransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.3.9 virtual long long activemq::commands::LocalTransactionId::getValue () const [virtual]

6.320.3.10 virtual bool activemq::commands::LocalTransactionId::operator< (const LocalTransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.3.11 LocalTransactionId& activemq::commands::LocalTransactionId::operator= (const LocalTransactionId & *other*)

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.3.12 virtual bool activemq::commands::LocalTransactionId::operator== (const LocalTransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.3.13 virtual void activemq::commands::LocalTransactionId::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.320.3.14 virtual void activemq::commands::LocalTransactionId::setValue (long long *value*) [virtual]

6.320.3.15 virtual std::string activemq::commands::LocalTransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.320.4 Field Documentation

6.320.4.1 Pointer<ConnectionId> activemq::commands::LocalTransactionId::connectionId [protected]

6.320.4.2 const unsigned char activemq::commands::LocalTransactionId::ID_LOCALTRANSACTIONID = 111 [static]

6.320.4.3 long long activemq::commands::LocalTransactionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**LocalTransactionId.h**

6.321 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdM Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1604).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual ~**LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.321.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1604). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.321.2 Constructor & Destructor Documentation

6.321.2.1 activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::LocalTransac () [inline]

6.321.2.2 virtual activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::~LocalTransa () [inline, virtual]

6.321.3 Member Function Documentation

6.321.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.321.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.321.3.3 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2586).

6.321.3.4 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2586).

6.321.3.5 virtual int activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2587).

6.321.3.6 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2587).

6.321.3.7 virtual void activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2588).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**LocalTransactionIdMarshaller.h**

6.322 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdM Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1608).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/LocalTransactionIdMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual ~**LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.322.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1608). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.322.2 Constructor & Destructor Documentation

6.322.2.1 activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::LocalTransac () [inline]

6.322.2.2 virtual activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::~LocalTransa () [inline, virtual]

6.322.3 Member Function Documentation

6.322.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.322.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.322.3.3 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582).

6.322.3.4 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582).

6.322.3.5 virtual int activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2583).

6.322.3.6 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2583).

6.322.3.7 virtual void activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2584).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**LocalTransactionIdMarshaller.h**

6.323 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdM Class Reference

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1612).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/LocalTransactionIdMarshaller.h>`Inherit diagram for activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller:

Public Member Functions

- **LocalTransactionIdMarshaller** ()
- virtual ~**LocalTransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.323.1 Detailed Description

Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1612). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.323.2 Constructor & Destructor Documentation

6.323.2.1 activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::LocalTransac () [inline]

6.323.2.2 virtual activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::~LocalTransa () [inline, virtual]

6.323.3 Member Function Documentation

6.323.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.323.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::getDataStructu () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.323.3.3 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2578).

6.323.3.4 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::looseUnmarsha (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2578).

6.323.3.5 virtual int activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2579).

6.323.3.6 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2579).

6.323.3.7 virtual void activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2580).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**LocalTransactionIdMarshaller.h**

6.324 decaf::util::concurrent::Lock Class Reference

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

`#include <src/main/decaf/util/concurrent/Lock.h>`

Public Member Functions

- **Lock** (**Synchronizable** *object, const bool intiallyLocked=true)

  *Constructor - initializes the object member and **locks** (p. 119) the object if desired.*

- virtual ~**Lock** ()

  Destructor - Unlocks the object if it is locked.

- void **lock** ()

  Locks the object.

- void **unlock** ()

  Unlocks the object.

- bool **isLocked** () const

  Indicates whether or not the object is locked.

6.324.1 Detailed Description

A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Author:

Nathan Mittler

6.324.2 Constructor & Destructor Documentation

6.324.2.1 decaf::util::concurrent::Lock::Lock (Synchronizable * *object*, const bool *intiallyLocked* = `true`) `[inline]`

Constructor - initializes the object member and **locks** (p. 119) the object if desired.

Parameters:

object The sync object to control

intiallyLocked If true, the object will automatically be locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and lock().

6.324.2.2 virtual decaf::util::concurrent::Lock::~Lock () [inline, virtual]

Destructor - Unlocks the object if it is locked.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

6.324.3 Member Function Documentation

6.324.3.1 bool decaf::util::concurrent::Lock::isLocked () const [inline]

Indicates whether or not the object is locked.

Returns:

true if the object is locked, otherwise false.

6.324.3.2 void decaf::util::concurrent::Lock::lock () [inline]

Locks the object.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::lock().

Referenced by Lock().

6.324.3.3 void decaf::util::concurrent::Lock::unlock () [inline]

Unlocks the object.

References DECAF_CATCH_RETHROW, DECAF_CATCHALL_THROW, and decaf::util::concurrent::Synchronizable::unlock().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Lock.h**

6.325 decaf::util::concurrent::locks::Lock Class Reference

Lock (p. 1618) implementations provide more extensive locking operations than can be obtained using synchronized statements.

`#include <src/main/decaf/util/concurrent/locks/Lock.h>`

Public Member Functions

- virtual ~**Lock** ()
- virtual void **lock** ()=0

  Acquires the lock.

- virtual void **lockInterruptibly** ()=0 throw (decaf::lang::exceptions::InterruptedException)

  Acquires the lock unless the current thread is interrupted.

- virtual bool **tryLock** ()=0

  Acquires the lock only if it is free at the time of invocation.

- virtual bool **tryLock** (long long time, const **TimeUnit** &unit)=0 throw (decaf::lang::exceptions::InterruptedException)

  Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted.

- virtual void **unlock** ()=0

  Releases the lock.

- virtual **Condition** * **newCondition** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  *Returns a new **Condition** (p. 932) instance that is bound to this **Lock** (p. 1618) instance.*

6.325.1 Detailed Description

Lock (p. 1618) implementations provide more extensive locking operations than can be obtained using synchronized statements. They allow more flexible structuring, may have quite different properties, and may support multiple associated **Condition** (p. 932) objects.

A lock is a tool for controlling access to a shared resource by multiple threads. Commonly, a lock provides exclusive access to a shared resource: only one thread at a time can acquire the lock and all access to the shared resource requires that the lock be acquired first. However, some **locks** (p. 119) may allow **concurrent** (p. 116) access to a shared resource, such as the read lock of a **ReadWriteLock** (p. 2170).

While the scoping mechanism for synchronized statements makes it much easier easier to program with monitor **locks** (p. 119), and helps avoid many common programming errors involving **locks** (p. 119), there are occasions where you need to work with **locks** (p. 119) in a more flexible way. For example, some algorithms for traversing concurrently accessed data structures require the use of "hand-over-hand" or "chain locking": you acquire the lock of node A, then node B, then release A and acquire C, then release B and acquire D and so on. Implementations of the **Lock**

(p. 1618) interface enable the use of such techniques by allowing a lock to be acquired and released in different scopes, and allowing multiple **locks** (p. 119) to be acquired and released in any order.

With this increased flexibility comes additional responsibility. The absence of block-structured locking removes the automatic release of **locks** (p. 119) that occurs with synchronized statements. In most cases, the following idiom should be used:

Lock (p. 1618) l = ...; l.lock(); try { // access the resource protected by this lock } catch(...) { l.unlock(); }

When locking and unlocking occur in different scopes, care must be taken to ensure that all code that is executed while the lock is held is protected by try-catch ensure that the lock is released when necessary.

Lock (p. 1618) implementations provide additional functionality over the use of synchronized methods and statements by providing a non-blocking attempt to acquire a lock (**tryLock()** (p. 1621)), an attempt to acquire the lock that can be interrupted (**lockInterruptibly()** (p. 1620), and an attempt to acquire the lock that can timeout (tryLock(long, TimeUnit)).

Note that **Lock** (p. 1618) instances are just normal objects and can themselves be used as the target in a synchronized statement.

The three forms of lock acquisition (interruptible, non-interruptible, and timed) may differ in their performance characteristics, ordering guarantees, or other implementation qualities. Further, the ability to interrupt the ongoing acquisition of a lock may not be available in a given **Lock** (p. 1618) class. Consequently, an implementation is not required to define exactly the same guarantees or semantics for all three forms of lock acquisition, nor is it required to support interruption of an ongoing lock acquisition. An implementation is required to clearly document the semantics and guarantees provided by each of the locking methods. It must also obey the interruption semantics as defined in this interface, to the extent that interruption of lock acquisition is supported: which is either totally, or only on method entry.

As interruption generally implies cancellation, and checks for interruption are often infrequent, an implementation can favor responding to an interrupt over normal method return. This is true even if it can be shown that the interrupt occurred after another action may have unblocked the thread. An implementation should document this behavior.

Since:

1.0

6.325.2 Constructor & Destructor Documentation

6.325.2.1 virtual decaf::util::concurrent::locks::Lock::~Lock () [inline, virtual]

6.325.3 Member Function Documentation

6.325.3.1 virtual void decaf::util::concurrent::locks::Lock::lock () [pure virtual]

Acquires the lock. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until the lock has been acquired.

Implementation Considerations

A **Lock** (p. 1618) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1618) implementation.

6.325.3.2 virtual void decaf::util::concurrent::locks::Lock::lockInterruptibly () throw (decaf::lang::exceptions::InterruptedException) [pure virtual]

Acquires the lock unless the current thread is interrupted. Acquires the lock if it is available and returns immediately.

If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of two things happens:

* The lock is acquired by the current thread; or * Some other thread interrupts the current thread, and interruption of lock acquisition is supported.

If the current thread:

* has its interrupted status set on entry to this method; or * is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return.

A **Lock** (p. 1618) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1618) implementation.

Exceptions:

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported).

6.325.3.3 virtual Condition* decaf::util::concurrent::locks::Lock::newCondition () throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Returns a new **Condition** (p. 932) instance that is bound to this **Lock** (p. 1618) instance. Before waiting on the condition the lock must be held by the current thread. A call to **Condition.await()** (p. 934) will atomically release the lock before waiting and re-acquire the lock before the wait returns.

Implementation Considerations

The exact operation of the **Condition** (p. 932) instance depends on the **Lock** (p. 1618) implementation and must be documented by that implementation.

Returns:

A new **Condition** (p. 932) instance for this **Lock** (p. 1618) instance the caller must delete the returned **Condition** (p. 932) object when done with it.

Exceptions:

UnsupportedOperationException if this **Lock** (p. 1618) implementation does not support conditions

6.325.3.4 virtual bool decaf::util::concurrent::locks::Lock::tryLock (long long *time*, const TimeUnit & *unit*) throw (decaf::lang::exceptions::InterruptedException) [pure virtual]

Acquires the lock if it is free within the given waiting time and the current thread has not been interrupted. If the lock is available this method returns immediately with the value true. If the lock is not available then the current thread becomes disabled for thread scheduling purposes and lies dormant until one of three things happens:

∗ The lock is acquired by the current thread; or ∗ Some other thread interrupts the current thread, and interruption of lock acquisition is supported; or ∗ The specified waiting time elapses

If the lock is acquired then the value true is returned.

If the current thread:

∗ has its interrupted status set on entry to this method; or ∗ is interrupted while acquiring the lock, and interruption of lock acquisition is supported,

then InterruptedException is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then the value false is returned. If the time is less than or equal to zero, the method will not wait at all.

Implementation Considerations

The ability to interrupt a lock acquisition in some implementations may not be possible, and if possible may be an expensive operation. The programmer should be aware that this may be the case. An implementation should document when this is the case.

An implementation can favor responding to an interrupt over normal method return, or reporting a timeout.

A **Lock** (p. 1618) implementation may be able to detect erroneous use of the lock, such as an invocation that would cause deadlock, and may throw an (unchecked) exception in such circumstances. The circumstances and the exception type must be documented by that **Lock** (p. 1618) implementation.

Parameters:

time the maximum time to wait for the lock

unit the time unit of the time argument

Returns:

true if the lock was acquired and false if the waiting time elapsed before the lock was acquired

Exceptions:

InterruptedException if the current thread is interrupted while acquiring the lock (and interruption of lock acquisition is supported)

6.325.3.5 virtual bool decaf::util::concurrent::locks::Lock::tryLock () [pure virtual]

Acquires the lock only if it is free at the time of invocation. Acquires the lock if it is available and returns immediately with the value true. If the lock is not available then this method will return immediately with the value false.

A typical usage idiom for this method would be:

Lock (p. 1618) lock = ...; if (lock.tryLock()) { try { // manipulate protected state } catch(...) { lock.unlock(); } } else { // perform alternative actions }

This usage ensures that the lock is unlocked if it was acquired, and doesn't try to unlock if the lock was not acquired.

Returns:

true if the lock was acquired and false otherwise

6.325.3.6 virtual void decaf::util::concurrent::locks::Lock::unlock () [pure virtual]

Releases the lock. Implementation Considerations

A **Lock** (p. 1618) implementation will usually impose restrictions on which thread can release a lock (typically only the holder of the lock can release it) and may throw an exception if the restriction is violated. Any restrictions and the exception type must be documented by that **Lock** (p. 1618) implementation.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**Lock.h**

6.326 decaf::util::logging::Logger Class Reference

`#include <src/main/decaf/util/logging/Logger.h>`

Public Member Functions

- **Logger** (const std::string &name, **Logger** *parent)

  *Creates a new instance of the **Logger** (p. 1623) with the given name and assign it the given parent logger.*

- virtual ~**Logger** ()
- virtual const std::string & **getName** () const

  *Gets the name of this **Logger** (p. 1623).*

- virtual void **addHandler** (**Handler** *handler) throw (lang::exceptions::IllegalArgumentException)

  *Add a log **Handler** (p. 1382) to receive **logging** (p. 120) messages.*

- virtual void **removeHandler** (**Handler** *handler)

  *Removes the specified **Handler** (p. 1382) and destroys it.*

- virtual void **setFilter** (**Filter** *filter)

  Gets a vector containing all the handlers that this class has been assigned to use.

- virtual const **Filter** * **getFilter** () const

  *Gets the **Filter** (p. 1314) object that this class is using.*

- virtual **Level getLevel** () const

  *Get the log Level that has been specified for this **Logger** (p. 1623).*

- virtual void **setLevel** (**Level** level)

  ***Set** (p. 2320) the log level specifying which message levels will be logged by this logger.*

- virtual bool **getUseParentHandlers** () const

  Discover whether or not this logger is sending its output to its parent logger.

- virtual void **setUseParentHandlers** (bool value)

  *pecify whether or not this logger should send its output to it's parent **Logger** (p. 1623).*

- virtual void **entry** (const std::string &blockName, const std::string &file, const int line)

  Logs an Block Enter message.

- virtual void **exit** (const std::string &blockName, const std::string &file, const int line)

  Logs an Block Exit message.

- virtual void **debug** (const std::string &file, const int line, const std::string functionName, const std::string &message)

  Log a Debug Level Log.

- virtual void **info** (const std::string &file, const int line, const std::string functionName, const std::string &message)

  Log a info Level Log.

- virtual void **warn** (const std::string &file, const int line, const std::string functionName, const std::string &message)

  Log a warn Level Log.

- virtual void **error** (const std::string &file, const int line, const std::string fnctionName, const std::string &message)

  Log a error Level Log.

- virtual void **fatal** (const std::string &file, const int line, const std::string functionName, const std::string &message)

  Log a fatal Level Log.

- virtual bool **isLoggable** (**Level** level) const

  Log a Throw Message.

- virtual void **log** (**LogRecord** &record)

  *Log a **LogRecord** (p. 1645).*

- virtual void **log** (**Level** level, const std::string &message)

  Log a message, with no arguments.

- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message,...)

  Log a message, with the list of params that is formatted into the message string.

- virtual void **log** (**Level** level, const std::string &file, const int line, const std::string &message, **lang::Exception** &ex)

  Log a message, with associated Throwable information.

Static Public Member Functions

- static **Logger** * **getAnonymousLogger** ()

  Creates an anonymous logger.

- static **Logger** * **getLogger** (const std::string &name)

  Find or create a logger for a named subsystem.

6.326.1 Constructor & Destructor Documentation

6.326.1.1 decaf::util::logging::Logger::Logger (const std::string & *name*, Logger * *parent*)

Creates a new instance of the **Logger** (p. 1623) with the given name and assign it the given parent logger. The logger will be initially configured with a null Level and with useParentHandlers true.

Parameters:

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as java.net or javax.swing. It may be null for anonymous Loggers.

parent logger that is this one's parent

6.326.1.2 virtual decaf::util::logging::Logger::~Logger () [virtual]

6.326.2 Member Function Documentation

6.326.2.1 virtual void decaf::util::logging::Logger::addHandler (Handler * *handler*) throw (lang::exceptions::IllegalArgumentException) [virtual]

Add a log **Handler** (p. 1382) to receive **logging** (p. 120) messages. By default, Loggers also send their output to their parent logger. Typically the root **Logger** (p. 1623) is configured with a set of Handlers that essentially act as default handlers for all loggers.

Parameters:

handler A Logging **Handler** (p. 1382)

Exceptions:

IllegalArgumentException

6.326.2.2 virtual void decaf::util::logging::Logger::debug (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a Debug Level Log. If the logger is currently enabled for the DEBUG message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

6.326.2.3 virtual void decaf::util::logging::Logger::entry (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Enter message. This is a convenience method that is used to tag a block enter, a log record with the class name function name and the string Entering is logged at the DEBUG log level.

Parameters:

blockName source block name

file source file name

line source line name

6.326.2.4 virtual void decaf::util::logging::Logger::error (const std::string & *file*, const int *line*, const std::string *fnctionName*, const std::string & *message*) [virtual]

Log a error Level Log. If the logger is currently enabled for the error message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

fnctionName name of the function that logged this

message the message to log

6.326.2.5 virtual void decaf::util::logging::Logger::exit (const std::string & *blockName*, const std::string & *file*, const int *line*) [virtual]

Logs an Block Exit message. This is a convenience method that is used to tag a block exit, a log record with the class name function name and the string Exiting is logged at the DEBUG log level.

Parameters:

blockName source block name

file source file name

line source line name

6.326.2.6 virtual void decaf::util::logging::Logger::fatal (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a fatal Level Log. If the logger is currently enabled for the fatal message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

6.326.2.7 static Logger* decaf::util::logging::Logger::getAnonymousLogger () [static]

Creates an anonymous logger. The newly created **Logger** (p. 1623) is not registered in the **LogManager** (p. 1640) namespace. There will be no access checks on updates to the logger. Even although the new logger is anonymous, it is configured to have the root logger ("") as its parent. This means that by default it inherits its effective level and handlers from the root logger.

The caller is responsible for destroying the returned logger.

Returns:

Newly created anonymous logger

6.326.2.8 virtual const Filter* decaf::util::logging::Logger::getFilter () const [inline, virtual]

Gets the **Filter** (p. 1314) object that this class is using.

Returns:

the **Filter** (p. 1314) in use, can be null

6.326.2.9 virtual Level decaf::util::logging::Logger::getLevel () const [inline, virtual]

Get the log Level that has been specified for this **Logger** (p. 1623). The result may be the Null level, which means that this logger's effective level will be inherited from its parent.

Returns:

the level that is currently set

6.326.2.10 static Logger* decaf::util::logging::Logger::getLogger (const std::string & *name*) [static]

Find or create a logger for a named subsystem. If a logger has already been created with the given name it is returned. Otherwise a new logger is created.

If a new logger is created its log level will be configured based on the **LogManager** (p. 1640) and it will configured to also send **logging** (p. 120) output to its parent loggers Handlers. It will be registered in the **LogManager** (p. 1640) global namespace.

Parameters:

name - A name for the logger. This should be a dot-separated name and should normally be based on the package name or class name of the subsystem, such as **cms** (p. 91) or **activemq.core.ActiveMQConnection** (p. 190)

Returns:

a suitable logger.

6.326.2.11 virtual const std::string& decaf::util::logging::Logger::getName () const [inline, virtual]

Gets the name of this **Logger** (p. 1623).

Returns:

logger name

6.326.2.12 virtual bool decaf::util::logging::Logger::getUseParentHandlers () const [inline, virtual]

Discover whether or not this logger is sending its output to its parent logger.

Returns:

true if using Parent Handlers

6.326.2.13 virtual void decaf::util::logging::Logger::info (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a info Level Log. If the logger is currently enabled for the info message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

6.326.2.14 virtual bool decaf::util::logging::Logger::isLoggable (Level *level*) const [virtual]

Log a Throw Message. If the logger is currently enabled for the Throwing message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log virtual void throwing(const std::string& file, const int line, const std::string functionName, const std::string& message); Check if a message of the given level would actually be logged by this logger. This check is based on the Loggers effective level, which may be inherited from its parent.

level - a message **logging** (p. 120) level

Returns:

true if the given message level is currently being logged.

6.326.2.15 virtual void decaf::util::logging::Logger::log (Level *level*, const std::string & *file*, const int *line*, const std::string & *message*, lang::Exception & *ex*) [virtual]

Log a message, with associated Throwable information. If the logger is currently enabled for the given message level then the given arguments are stored in a **LogRecord** (p. 1645) which is

forwarded to all registered output handlers. Note that the thrown argument is stored in the **LogRecord** (p. 1645) thrown property, rather than the **LogRecord** (p. 1645) parameters property. Thus is it processed specially by output Formatters and is not treated as a formatting parameter to the **LogRecord** (p. 1645) message property.

Parameters:

level the Level to log at.

file File that the message was logged in.

line the line number where the message was logged at.

message the message to log.

ex the Exception to log

6.326.2.16 virtual void decaf::util::logging::Logger::log (Level *level*, const std::string & *file*, const int *line*, const std::string & *message*, ...) [virtual]

Log a message, with the list of params that is formatted into the message string. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects

Parameters:

level the Level to log at

file the message to log

line the line in the file

... variable length argument to format the message string.

6.326.2.17 virtual void decaf::util::logging::Logger::log (Level *level*, const std::string & *message*) [virtual]

Log a message, with no arguments. If the logger is currently enabled for the given message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects

Parameters:

level the Level to log at

message the message to log

6.326.2.18 virtual void decaf::util::logging::Logger::log (LogRecord & *record*) [virtual]

Log a **LogRecord** (p. 1645). All the other **logging** (p. 120) methods in this class call through this method to actually perform any **logging** (p. 120). Subclasses can override this single method to capture all log activity.

Parameters:

record - the **LogRecord** (p. 1645) to be published

6.326.2.19 virtual void decaf::util::logging::Logger::removeHandler (Handler * *handler*) [virtual]

Removes the specified **Handler** (p. 1382) and destroys it. Returns silently if the given **Handler** (p. 1382) is not found.

Parameters:

handler The **Handler** (p. 1382) to remove

6.326.2.20 virtual void decaf::util::logging::Logger::setFilter (Filter * *filter*) [virtual]

Gets a vector containing all the handlers that this class has been assigned to use.

Returns:

a list of handlers that are used by this logger **Set** (p. 2320) a filter to control output on this **Logger** (p. 1623).

After passing the initial "level" check, the **Logger** (p. 1623) will call this **Filter** (p. 1314) to check if a log record should really be published.

The caller releases ownership of this filter to this logger

Parameters:

filter to use, can be null

6.326.2.21 virtual void decaf::util::logging::Logger::setLevel (Level *level*) [inline, virtual]

Set (p. 2320) the log level specifying which message levels will be logged by this logger. Message levels lower than this value will be discarded. The level value Level.OFF can be used to turn off **logging** (p. 120).

If the new level is the Null Level, it means that this node should inherit its level from its nearest ancestor with a specific (non-null) level value.

Parameters:

level new Level value

6.326.2.22 virtual void decaf::util::logging::Logger::setUseParentHandlers (bool *value*) [inline, virtual]

pecify whether or not this logger should send its output to it's parent **Logger** (p. 1623). This means that any LogRecords will also be written to the parent's Handlers, and potentially to its parent, recursively up the namespace.

Parameters:

value True is output is to be writen to the parent

6.326.2.23 virtual void decaf::util::logging::Logger::warn (const std::string & *file*, const int *line*, const std::string *functionName*, const std::string & *message*) [virtual]

Log a warn Level Log. If the logger is currently enabled for the warn message level then the given message is forwarded to all the registered output **Handler** (p. 1382) objects.

Parameters:

file the file name where the log was generated

line the line number where the log was generated

functionName name of the function that logged this

message the message to log

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**Logger.h**

6.327 decaf::util::logging::LoggerHierarchy Class Reference

`#include <src/main/decaf/util/logging/LoggerHierarchy.h>`

Public Member Functions

- **LoggerHierarchy** ()
- virtual ~**LoggerHierarchy** ()

6.327.1 Constructor & Destructor Documentation

6.327.1.1 decaf::util::logging::LoggerHierarchy::LoggerHierarchy ()

6.327.1.2 virtual decaf::util::logging::LoggerHierarchy::~LoggerHierarchy () [virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LoggerHierarchy.h**

6.328 activemq::io::LoggingInputStream Class Reference

`#include <src/main/activemq/io/LoggingInputStream.h>`Inheritance diagram for activemq::io::LoggingInputStream:

Public Member Functions

- **LoggingInputStream** (**decaf::io::InputStream** ∗**inputStream**, bool **own**=false)

  Creates a DataInputStream that uses the specified underlying InputStream.

- virtual ∼**LoggingInputStream** ()
- virtual unsigned char **read** () throw (decaf::io::IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char ∗buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

6.328.1 Constructor & Destructor Documentation

6.328.1.1 activemq::io::LoggingInputStream::LoggingInputStream (decaf::io::InputStream ∗ *inputStream*, bool *own* = `false`)

Creates a DataInputStream that uses the specified underlying InputStream.

Parameters:

inputStream the InputStream instance to wrap.

own indicates if this class owns the wrapped string defaults to false.

6.328.1.2 virtual activemq::io::LoggingInputStream::∼LoggingInputStream () `[virtual]`

6.328.2 Member Function Documentation

6.328.2.1 virtual int activemq::io::LoggingInputStream::read (unsigned char ∗ *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) `[virtual]`

Reads an array of bytes from the buffer. Blocks until the requested number of bytes are available.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start at

bufferSize the size of the output buffer.

Returns:

The number of bytes read or -1 if EOF is detected

Exceptions:

IOException thrown if an error occurs.

NullPointerException if buffer is null

Reimplemented from **decaf::io::FilterInputStream** (p. 1319).

6.328.2.2 virtual unsigned char activemq::io::LoggingInputStream::read () throw (decaf::io::IOException) [virtual]

Reads a single byte from the buffer. Blocks until data is available.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Reimplemented from **decaf::io::FilterInputStream** (p. 1319).

The documentation for this class was generated from the following file:

- src/main/activemq/io/**LoggingInputStream.h**

6.329 activemq::io::LoggingOutputStream Class Reference

OutputStream filter that just logs the data being written.

`#include <src/main/activemq/io/LoggingOutputStream.h>`Inheritance diagram for activemq::io::LoggingOutputStream:

Public Member Functions

- **LoggingOutputStream** (OutputStream *next, bool **own**=false)

  Constructor.

- virtual ~**LoggingOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

6.329.1 Detailed Description

OutputStream filter that just logs the data being written.

6.329.2 Constructor & Destructor Documentation

6.329.2.1 activemq::io::LoggingOutputStream::LoggingOutputStream (OutputStream * *next*, bool *own* = `false`)

Constructor.

Parameters:

next The OutputStream to wrap an write logs to.

own If true, this object will control the lifetime of the output stream that it encapsulates.

6.329.2.2 virtual activemq::io::LoggingOutputStream::~LoggingOutputStream () `[virtual]`

6.329.3 Member Function Documentation

6.329.3.1 virtual void activemq::io::LoggingOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) `[virtual]`

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position in the buffer to start at

len The number of bytes from the buffer to be written.

Exceptions:

IOException thrown if an error occurs.

NullPointerException if buffer is null.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1326).

6.329.3.2 virtual void activemq::io::LoggingOutputStream::write (unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Reimplemented from **decaf::io::FilterOutputStream** (p. 1327).

The documentation for this class was generated from the following file:

- src/main/activemq/io/**LoggingOutputStream.h**

6.330 activemq::transport::logging::LoggingTransport Class Reference

A **transport** (p. 67) filter that logs **commands** (p. 59) as they are sent/received.

`#include <src/main/activemq/transport/logging/LoggingTransport.h>`Inheritance diagram for activemq::transport::logging::LoggingTransport:

Public Member Functions

- **LoggingTransport** (const **Pointer**< **Transport** > &**next**)

  Constructor.

- virtual ~**LoggingTransport** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)

  Event handler for the receipt of a command.

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Not supported by this class - throws an exception.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Not supported by this class - throws an exception.

6.330.1 Detailed Description

A **transport** (p. 67) filter that logs **commands** (p. 59) as they are sent/received.

6.330.2 Constructor & Destructor Documentation

6.330.2.1 activemq::transport::logging::LoggingTransport::LoggingTransport (const Pointer< Transport > & *next*)

Constructor.

Parameters:

next - the next **Transport** (p. 2608) in the chain

6.330.2.2 virtual activemq::transport::logging::LoggingTransport::~LoggingTransport () [inline, virtual]

6.330.3 Member Function Documentation

6.330.3.1 virtual void activemq::transport::logging::LoggingTransport::onCommand (const Pointer< Command > & *command*) [virtual]

Event handler for the receipt of a command.

Parameters:

command - the received command object.

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.330.3.2 virtual void activemq::transport::logging::LoggingTransport::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.330.3.3 virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

timeout the time to wait for a response.

Exceptions:

UnsupportedOperationException.

Reimplemented from **activemq::transport::TransportFilter** (p. 2621).

6.330.3.4 virtual Pointer<Response> activemq::transport::logging::LoggingTransport::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

Exceptions:

UnsupportedOperationException.

Reimplemented from **activemq::transport::TransportFilter** (p. 2622).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/logging/**LoggingTransport.h**

6.331 decaf::util::logging::LogManager Class Reference

There is a single global **LogManager** (p. 1640) object that is used to maintain a set of shared state about Loggers and log services.

`#include <src/main/decaf/util/logging/LogManager.h>`

Public Member Functions

- virtual ~**LogManager** ()
- virtual void **setProperties** (const **util::Properties** &properties)

  *Sets the **Properties** (p. 2140) this **LogManager** (p. 1640) should use to configure its loggers.*

- virtual const **util::Properties** & **getProperties** () const

  *Gets a reference to the Logging **Properties** (p. 2140) used by this logger.*

- virtual std::string **getProperty** (const std::string &name)

  *Gets the value of a named property of this **LogManager** (p. 1640).*

- virtual void **addPropertyChangeListener** (PropertyChangeListener *listener)

  *Adds a change listener for **LogManager** (p. 1640) **Properties** (p. 2140), adding the same instance of a change event listener does nothing.*

- virtual void **removePropertyChangeListener** (PropertyChangeListener *listener)

  *Removes a properties change listener from the **LogManager** (p. 1640).*

- virtual **Logger** * **getLogger** (const std::string &name)

  *Retrieves or creates a new **Logger** (p. 1623) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.*

- virtual int **getLoggerNames** (const std::vector< std::string > &names)

  *Gets a list of known **Logger** (p. 1623) Names from this Manager.*

Static Public Member Functions

- static **LogManager** * **getInstance** ()

  Get the singleton instance.

- static void **returnInstance** ()

  Returns a Checked out instance of this Manager.

- static void **destroy** ()

  *Forcefully Delete the Instance of this **LogManager** (p. 1640) even if there are outstanding references.*

Protected Member Functions

- **LogManager** ()

  Constructor, hidden to protect against direct instantiation.

- **LogManager** (const **LogManager** &manager)

  Copy Constructo.

- void **operator**= (const **LogManager** &manager)

  Assignment operator.

6.331.1 Detailed Description

There is a single global **LogManager** (p. 1640) object that is used to maintain a set of shared state about Loggers and log services. This **LogManager** (p. 1640) object:

* Manages a hierarchical namespace of **Logger** (p. 1623) objects. All named Loggers are stored in this namespace. * Manages a set of **logging** (p. 120) control properties. These are simple key-value pairs that can be used by Handlers and other **logging** (p. 120) objects to configure themselves.

The global **LogManager** (p. 1640) object can be retrieved using LogManager.getLogManager(). The **LogManager** (p. 1640) object is created during class initialization and cannot subsequently be changed.

TODO* By default, the **LogManager** (p. 1640) reads its initial configuration from a properties file "lib/logging.properties" in the JRE directory. If you edit that property file you can change the default **logging** (p. 120) configuration for all uses of that JRE.

In addition, the **LogManager** (p. 1640) uses two optional system properties that allow more control over reading the initial configuration:

* "decaf.logger.config.class" * "decaf.logger.config.file"

These two properties may be set via the Preferences API, or as command line property definitions to the "java" command, or as system property definitions passed to JNI_CreateJavaVM.

If the "java.util.logging.config.class" property is set, then the property value is treated as a class name. The given class will be loaded, an object will be instantiated, and that object's constructor is responsible for reading in the initial configuration. (That object may use other system properties to control its configuration.) The alternate configuration class can use readConfiguration(InputStream) to define properties in the **LogManager** (p. 1640).

If "java.util.logging.config.class" property is not set, then the "java.util.logging.config.file" system property can be used to specify a properties file (in java.util.Properties format). The initial **logging** (p. 120) configuration will be read from this file.

If neither of these properties is defined then, as described above, the **LogManager** (p. 1640) will read its initial configuration from a properties file "lib/logging.properties" in the JRE directory.

The properties for loggers and Handlers will have names starting with the dot-separated name for the handler or logger. ***TODO****

The global **logging** (p. 120) properties may include:

* A property "handlers". This defines a whitespace separated list of class names for handler classes to load and register as handlers on the root **Logger** (p. 1623) (the **Logger** (p. 1623) named ""). Each class name must be for a **Handler** (p. 1382) class which has a default constructor. Note that

these Handlers may be created lazily, when they are first used. * A property "<logger>.handlers". This defines a whitespace or comma separated list of class names for handlers classes to load and register as handlers to the specified logger. Each class name must be for a **Handler** (p. 1382) class which has a default constructor. Note that these Handlers may be created lazily, when they are first used. * A property "<logger>.useParentHandlers". This defines a boolean value. By default every logger calls its parent in addition to handling the **logging** (p. 120) message itself, this often result in messages being handled by the root logger as well. When setting this property to false a **Handler** (p. 1382) needs to be configured for this logger otherwise no **logging** (p. 120) messages are delivered. * A property "config". This property is intended to allow arbitrary configuration code to be run. The property defines a whitespace separated list of class names. A new instance will be created for each named class. The default constructor of each class may execute arbitrary code to update the **logging** (p. 120) configuration, such as setting logger levels, adding handlers, adding filters, etc.

Loggers are organized into a naming hierarchy based on their dot separated names. Thus "a.b.c" is a child of "a.b", but "a.b1" and a.b2 are peers.

All properties whose names end with ".level" are assumed to define log levels for Loggers. Thus "foo.level" defines a log level for the logger called "foo" and (recursively) for any of its children in the naming hierarchy. Log Levels are applied in the order they are defined in the properties file. Thus level settings for child nodes in the tree should come after settings for their parents. The property name ".level" can be used to set the level for the root of the tree.

All methods on the **LogManager** (p. 1640) object are multi-thread safe.

6.331.2 Constructor & Destructor Documentation

6.331.2.1 virtual decaf::util::logging::LogManager::~LogManager () [virtual]

6.331.2.2 decaf::util::logging::LogManager::LogManager () [inline, protected]

Constructor, hidden to protect against direct instantiation.

6.331.2.3 decaf::util::logging::LogManager::LogManager (const LogManager & *manager*) [protected]

Copy Constructo.

Parameters:

manager the Manager to copy

6.331.3 Member Function Documentation

6.331.3.1 virtual void decaf::util::logging::LogManager::addPropertyChangeListener (PropertyChangeListener * *listener*) [virtual]

Adds a change listener for **LogManager** (p. 1640) **Properties** (p. 2140), adding the same instance of a change event listener does nothing.

Parameters:

listener a PropertyChangeListener

6.331.3.2 static void decaf::util::logging::LogManager::destroy () [static]

Forcefully Delete the Instance of this **LogManager** (p. 1640) even if there are outstanding references.

6.331.3.3 static LogManager* decaf::util::logging::LogManager::getInstance () [static]

Get the singleton instance.

Returns:

Pointer to an instance of the Log Manager

6.331.3.4 virtual Logger* decaf::util::logging::LogManager::getLogger (const std::string & *name*) [virtual]

Retrieves or creates a new **Logger** (p. 1623) using the name specified a new logger inherits the configuration of the logger's parent if there is no configuration data for the logger.

Parameters:

name The name of the **Logger** (p. 1623).

6.331.3.5 virtual int decaf::util::logging::LogManager::getLoggerNames (const std::vector< std::string > & *names*) [virtual]

Gets a list of known **Logger** (p. 1623) Names from this Manager.

Parameters:

names STL Vector to hold string logger names

Returns:

names count of how many loggers were inserted

6.331.3.6 virtual const util::Properties& decaf::util::logging::LogManager::getProperties () const [inline, virtual]

Gets a reference to the Logging **Properties** (p. 2140) used by this logger.

Returns:

The **Logger** (p. 1623) **Properties** (p. 2140) Pointer

6.331.3.7 virtual std::string decaf::util::logging::LogManager::getProperty (const std::string & *name*) [inline, virtual]

Gets the value of a named property of this **LogManager** (p. 1640).

Parameters:

name of the Property to retrieve

Returns:

the value of the property

References decaf::util::Properties::getProperty().

6.331.3.8 void decaf::util::logging::LogManager::operator= (const LogManager & *manager*) [protected]

Assignment operator.

Parameters:

manager the manager to assign from

6.331.3.9 virtual void decaf::util::logging::LogManager::removePropertyChangeListener (PropertyChangeListener * *listener*) [virtual]

Removes a properties change listener from the **LogManager** (p. 1640).

Parameters:

listener a PropertyChangeListener

6.331.3.10 static void decaf::util::logging::LogManager::returnInstance () [static]

Returns a Checked out instance of this Manager.

6.331.3.11 virtual void decaf::util::logging::LogManager::setProperties (const util::Properties & *properties*) [virtual]

Sets the **Properties** (p. 2140) this **LogManager** (p. 1640) should use to configure its loggers. Once set a properties change event is fired.

Parameters:

properties Pointer to read the configuration from

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogManager.h**

6.332 decaf::util::logging::LogRecord Class Reference

`#include <src/main/decaf/util/logging/LogRecord.h>`

Public Member Functions

- **LogRecord** ()
- virtual ~**LogRecord** ()
- **Level getLevel** () const

  Get Level of this log record.

- void **setLevel** (**Level** value)

  Set *(p. 2320) the Level of this Log Record.*

- const std::string & **getLoggerName** () const

  Gets the Source Logger's Name.

- void **setLoggerName** (const std::string &loggerName)

  Sets the Source Logger's Name.

- const std::string & **getSourceFile** () const

  Gets the Source Log File name.

- void **setSourceFile** (const std::string &sourceFile)

  Sets the Source Log File Name.

- unsigned long **getSourceLine** () const

  Gets the Source Log line number.

- void **setSourceLine** (long sourceLine)

  Sets the Source Log line number.

- const std::string & **getMessage** () const

  Gets the Message to be Logged.

- void **setMessage** (const std::string &message)

  Sets the Message to be Logged.

- const std::string & **getSourceFunction** () const

  Gets the name of the function where this log was logged.

- void **setSourceFunction** (const std::string &functionName)

  Sets the name of the function where this log was logged.

- unsigned long **getTimestamp** () const

  Gets the time in mills that this message was logged.

- void **setTimestamp** (long timeStamp)

  Sets the time in mills that this message was logged.

- unsigned long **getTreadId** () const
  Gets the Thread Id where this Log was created.
- void **setTreadId** (long threadId)
  Sets the Thread Id where this Log was created.

6.332.1 Constructor & Destructor Documentation

6.332.1.1 decaf::util::logging::LogRecord::LogRecord () [inline]

6.332.1.2 virtual decaf::util::logging::LogRecord::~LogRecord () [inline, virtual]

6.332.2 Member Function Documentation

6.332.2.1 Level decaf::util::logging::LogRecord::getLevel () const [inline]

Get Level of this log record.

Returns:

Level enumeration value.

6.332.2.2 const std::string& decaf::util::logging::LogRecord::getLoggerName () const [inline]

Gets the Source Logger's Name.

Returns:

the source loggers name

6.332.2.3 const std::string& decaf::util::logging::LogRecord::getMessage () const [inline]

Gets the Message to be Logged.

Returns:

the source logger's message

Referenced by decaf::util::logging::SimpleFormatter::formatMessage().

6.332.2.4 const std::string& decaf::util::logging::LogRecord::getSourceFile () const [inline]

Gets the Source Log File name.

Returns:

the source loggers name

6.332.2.5 const std::string& decaf::util::logging::LogRecord::getSourceFunction () const [inline]

Gets the name of the function where this log was logged.

Returns:

the source logger's message

6.332.2.6 unsigned long decaf::util::logging::LogRecord::getSourceLine () const [inline]

Gets the Source Log line number.

Returns:

the source loggers line number

6.332.2.7 unsigned long decaf::util::logging::LogRecord::getTimestamp () const [inline]

Gets the time in mills that this message was logged.

Returns:

UTC time in milliseconds

6.332.2.8 unsigned long decaf::util::logging::LogRecord::getTreadId () const [inline]

Gets the Thread Id where this Log was created.

Returns:

the source loggers line number

6.332.2.9 void decaf::util::logging::LogRecord::setLevel (Level *value*) [inline]

Set (p. 2320) the Level of this Log Record.

Parameters:

value Level Enumeration Value

6.332.2.10 void decaf::util::logging::LogRecord::setLoggerName (const std::string & *loggerName*) [inline]

Sets the Source Logger's Name.

Parameters:

loggerName the source loggers name

6.332.2.11 void decaf::util::logging::LogRecord::setMessage (const std::string & *message*) [inline]

Sets the Message to be Logged.

Parameters:

message the source loggers message

6.332.2.12 void decaf::util::logging::LogRecord::setSourceFile (const std::string & *sourceFile*) [inline]

Sets the Source Log File Name.

Parameters:

sourceFile the source loggers name

6.332.2.13 void decaf::util::logging::LogRecord::setSourceFunction (const std::string & *functionName*) [inline]

Sets the name of the function where this log was logged.

Parameters:

functionName the source of the log

6.332.2.14 void decaf::util::logging::LogRecord::setSourceLine (long *sourceLine*) [inline]

Sets the Source Log line number.

Parameters:

sourceLine the source logger's line number

6.332.2.15 void decaf::util::logging::LogRecord::setTimestamp (long *timeStamp*) [inline]

Sets the time in mills that this message was logged.

Parameters:

timeStamp UTC Time in Milliseconds.

6.332.2.16 void decaf::util::logging::LogRecord::setTreadId (long *threadId*) [inline]

Sets the Thread Id where this Log was created.

Parameters:

threadId the source logger's line number

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogRecord.h**

6.333 decaf::util::logging::LogWriter Class Reference

`#include <src/main/decaf/util/logging/LogWriter.h>`

Public Member Functions

- **LogWriter** ()
- virtual ~**LogWriter** ()
- virtual void **log** (const std::string &file, const int line, const std::string &prefix, const std::string &message)

  Writes a message to the output destination.

- virtual void **log** (const std::string &message)

  Writes a message to the output destination.

Static Public Member Functions

- static **LogWriter** & **getInstance** ()

  Get the singleton instance.

- static void **returnInstance** ()

  Returns a Checked out instance of this Writer.

- static void **destroy** ()

  *Forcefully Delete the Instance of this **LogWriter** (p. 1650) even if there are outstanding references.*

6.333.1 Constructor & Destructor Documentation

6.333.1.1 decaf::util::logging::LogWriter::LogWriter ()

6.333.1.2 virtual decaf::util::logging::LogWriter::~LogWriter () [virtual]

6.333.2 Member Function Documentation

6.333.2.1 static void decaf::util::logging::LogWriter::destroy () [static]

Forcefully Delete the Instance of this **LogWriter** (p. 1650) even if there are outstanding references.

6.333.2.2 static LogWriter& decaf::util::logging::LogWriter::getInstance () [static]

Get the singleton instance.

6.333.2.3 virtual void decaf::util::logging::LogWriter::log (const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters:

message

6.333.2.4 virtual void decaf::util::logging::LogWriter::log (const std::string & *file*, const int *line*, const std::string & *prefix*, const std::string & *message*) [virtual]

Writes a message to the output destination.

Parameters:

file

line

prefix

message

6.333.2.5 static void decaf::util::logging::LogWriter::returnInstance () [static]

Returns a Checked out instance of this Writer.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**LogWriter.h**

6.334 decaf::lang::Long Class Reference

`#include <src/main/decaf/lang/Long.h>`Inheritance diagram for decaf::lang::Long:

Public Member Functions

- **Long** (long long value)
- **Long** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Long** ()
- virtual int **compareTo** (const **Long** &l) const

  *Compares this **Long** (p. 1652) instance with another.*

- bool **equals** (const **Long** &l) const
- virtual bool **operator==** (const **Long** &l) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Long** &l) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const long long &l) const

  *Compares this **Long** (p. 1652) instance with another.*

- bool **equals** (const long long &l) const
- virtual bool **operator==** (const long long &l) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const long long &l) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

Static Public Member Functions

- static int **bitCount** (long long value)

  Returns the number of one-bits in the two's complement binary representation of the specified int value.

- static **Long decode** (const std::string &value) throw (exceptions::NumberFormatException)

  *Decodes a String into a **Long** (*p. *1652)**.*

- static long long **highestOneBit** (long long value)

  Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value.

- static long long **lowestOneBit** (long long value)

  Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value.

- static int **numberOfLeadingZeros** (long long value)

  Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value.

- static int **numberOfTrailingZeros** (long long value)

  Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value.

- static long long **parseLong** (const std::string &value) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed decimal long.

- static long long **parseLong** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

  *Returns a **Long** (*p. *1652)** object holding the value extracted from the specified string when parsed with the radix given by the second argument.*

- static long long **reverseBytes** (long long value)

  Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

- static long long **reverse** (long long value)

  Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

- static long long **rotateLeft** (long long value, int distance)

  Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

- static long long **rotateRight** (long long value, int distance)

  Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

- static int **signum** (long long value)

Returns the signum function of the specified value.

- static std::string **toString** (long long value)

  Converts the long to a String representation.

- static std::string **toString** (long long value, int radix)
- static std::string **toHexString** (long long value)

  Returns a string representation of the integer argument as an unsigned integer in base 16.

- static std::string **toOctalString** (long long value)

  Returns a string representation of the long long argument as an unsigned long long in base 8.

- static std::string **toBinaryString** (long long value)

  Returns a string representation of the long long argument as an unsigned long long in base 2.

- static **Long valueOf** (long long value)

  *Returns a **Long** (p. 1652) instance representing the specified int value.*

- static **Long valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a **Long** (p. 1652) object holding the value given by the specified std::string.*

- static **Long valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

  *Returns a **Long** (p. 1652) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 64

  The size in bits of the primitive long long type.

- static const long long **MAX_VALUE** = (long long)0x7FFFFFFFFFFFFFFFLL

  The maximum value that the primitive type can hold.

- static const long long **MIN_VALUE** = (long long)0x8000000000000000LL

  The minimum value that the primitive type can hold.

6.334.1 Constructor & Destructor Documentation

6.334.1.1 decaf::lang::Long::Long (long long *value*)

Parameters:

value - the primitive long long to wrap

6.334.1.2 decaf::lang::Long::Long (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - the long long formated string to wrap

Exceptions:

NumberFormatException

6.334.1.3 virtual decaf::lang::Long::~Long () [inline, virtual]

6.334.2 Member Function Documentation

6.334.2.1 static int decaf::lang::Long::bitCount (long long *value*) [static]

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the population count.

Parameters:

value - the long long to count

Returns:

the number of one-bits in the two's complement binary representation of the specified long long value.

6.334.2.2 virtual unsigned char decaf::lang::Long::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.334.2.3 virtual int decaf::lang::Long::compareTo (const long long & *l*) const [virtual]

Compares this **Long** (p. 1652) instance with another.

Parameters:

l - the **Integer** (p. 1428) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< long long >** (p. 899).

6.334.2.4 virtual int decaf::lang::Long::compareTo (const Long & *l*) const [virtual]

Compares this **Long** (p. 1652) instance with another.

Parameters:

l - the **Long** (p. 1652) instance to be compared

Returns:

zero if this object represents the same integer value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

6.334.2.5 static Long decaf::lang::Long::decode (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Long** (p. 1652). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInteger method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Long** (p. 1652) object containing the decoded value

Exceptions:

NumberFomatException if the string is not formatted correctly.

6.334.2.6 virtual double decaf::lang::Long::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.334.2.7 bool decaf::lang::Long::equals (const long long & *l*) const [inline, virtual]

Parameters:

l - the **Long** (p. 1652) object to compare against.

Returns:

true if the two **Integer** (p. 1428) Objects have the same value.

Implements **decaf::lang::Comparable< long long >** (p. 900).

6.334.2.8 bool decaf::lang::Long::equals (const Long & *l*) const [inline]

Parameters:

l - the **Long** (p. 1652) object to compare against.

Returns:

true if the two **Integer** (p. 1428) Objects have the same value.

6.334.2.9 virtual float decaf::lang::Long::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.334.2.10 static long long decaf::lang::Long::highestOneBit (long long *value*) [static]

Returns an long long value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the long long to be inspected

Returns:

an long long value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.334.2.11 virtual int decaf::lang::Long::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.334.2.12 virtual long long decaf::lang::Long::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.334.2.13 static long long decaf::lang::Long::lowestOneBit (long long *value*) [static]

Returns an long long value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

Parameters:

value - the long long to be inspected

Returns:

an long long value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.

6.334.2.14 static int decaf::lang::Long::numberOfLeadingZeros (long long *value*) [static]

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

∗ floor(log2(x)) = 63 - numberOfLeadingZeros(x) ∗ ceil(log2(x)) = 64 - numberOfLeadingZeros(x - 1)

Parameters:

value - the long long to be inspected

Returns:

the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.334.2.15 static int decaf::lang::Long::numberOfTrailingZeros (long long *value*) [static]

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value. Returns 64 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Parameters:

value - the int to be inspected

Returns:

the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified long long value, or 64 if the value is equal to zero.

6.334.2.16 virtual bool decaf::lang::Long::operator< (const long long & *l*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 900).

6.334.2.17 virtual bool decaf::lang::Long::operator< (const Long & *l*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.334.2.18 virtual bool decaf::lang::Long::operator== (const long long & *l*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< long long >** (p. 901).

6.334.2.19 virtual bool decaf::lang::Long::operator== (const Long & *l*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

l - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.334.2.20 static long long decaf::lang::Long::parseLong (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Long** (p. 1652) object holding the value extracted from the specified string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long in the radix specified by the second argument, exactly as if the arguments were given to the parseLong(std::string, int) method. The result is a **Long** (p. 1652) object that represents the long long value specified by the string.

Parameters:

value - String to parse

radix - the base encoding of the string

Returns:

long long value

Exceptions:

NumberFormatException on invalid string value

6.334.2.21 static long long decaf::lang::Long::parseLong (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal long. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting long value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseLong(java.lang.String, int) method.

Note that the characters LL or ULL are not permitted to appear at the end of this string as would normally be permitted in a C++ program.

Parameters:

value - String to parse

Returns:

long long value

Exceptions:

NumberFormatException on invalid string value

6.334.2.22 static long long decaf::lang::Long::reverse (long long *value*) [static]

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified long long value.

Parameters:

value - the value whose bits are to be reversed

Returns:

the reversed bits long long.

6.334.2.23 static long long decaf::lang::Long::reverseBytes (long long *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified long long value.

Parameters:

value - the long long whose bytes we are to reverse

Returns:

the reversed long long.

6.334.2.24 static long long decaf::lang::Long::rotateLeft (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

Parameters:

value - the long long to be inspected

distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified value left by the specified number of bits.

6.334.2.25 static long long decaf::lang::Long::rotateRight (long long *value*, int *distance*) [static]

Returns the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

Parameters:

value - the long long to be inspected

distance - the number of bits to rotate

Returns:

the value obtained by rotating the two's complement binary representation of the specified value right by the specified number of bits.

6.334.2.26 virtual short decaf::lang::Long::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.334.2.27 static int decaf::lang::Long::signum (long long *value*) [static]

Returns the signum function of the specified value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

Parameters:

value - the long long to be inspected

Returns:

the signum function of the specified long long value.

6.334.2.28 static std::string decaf::lang::Long::toBinaryString (long long *value*) [static]

Returns a string representation of the long long argument as an unsigned long long in base 2. The unsigned long long value is the argument plus 2^32 if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' and '1' are used as binary digits.

Parameters:

value - the long long to be translated to a binary string

Returns:

the unsigned long long value as a binary string

6.334.2.29 static std::string decaf::lang::Long::toHexString (long long *value*) [static]

Returns a string representation of the integer argument as an unsigned integer in base 16. The unsigned integer value is the argument plus 2^32 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

If uppercase letters are desired, the toUpperCase() method may be called on the result:

Parameters:

value - the long long to be translated to an Octal string

Returns:

the unsigned long long value as a Octal string

6.334.2.30 static std::string decaf::lang::Long::toOctalString (long long *value*) [static]

Returns a string representation of the long long argument as an unsigned long long in base 8. The unsigned long long value is the argument plus 2^32 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0'; otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

Parameters:

value - the long long to be translated to an Octal string

Returns:

the unsigned long long value as a Octal string

6.334.2.31 static std::string decaf::lang::Long::toString (long long *value*, int *radix*) [static]

6.334.2.32 static std::string decaf::lang::Long::toString (long long *value*) [static]

Converts the long to a String representation.

Parameters:

value The long to convert to a std::string.

Returns:

string representation

6.334.2.33 std::string decaf::lang::Long::toString () const

Returns:

this **Long** (p. 1652) Object as a String Representation

6.334.2.34 static Long decaf::lang::Long::valueOf (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Long** (p. 1652) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed long long in the radix specified by the second argument, exactly as if the argument were given to the parseLong(std::string, int) method. The result is a **Long** (p. 1652) object that represents the long long value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Long** (p. 1652) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid long long.

6.334.2.35 static Long decaf::lang::Long::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Long** (p. 1652) object holding the value given by the specified std::string. The argument is interpreted as representing a signed decimal long long, exactly as if the argument were given to the parseLong(std::string) method. The result is a **Integer** (p. 1428) object that represents the long long value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Long** (p. 1652) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal long long.

6.334.2.36 static Long decaf::lang::Long::valueOf (long long *value*) [inline, static]

Returns a **Long** (p. 1652) instance representing the specified int value.

Parameters:

value - the long long to wrap

Returns:

the new **Integer** (p. 1428) object wrapping value.

6.334.3 Field Documentation

6.334.3.1 const long long decaf::lang::Long::MAX_VALUE = (long long)0x7FFFFFFFFFFFFFFFLL [static]

The maximum value that the primitive type can hold.

6.334.3.2 const long long decaf::lang::Long::MIN_VALUE = (long long)0x8000000000000000LL [static]

The minimum value that the primitive type can hold.

6.334.3.3 const int decaf::lang::Long::SIZE = 64 [static]

The size in bits of the primitive long long type.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Long.h**

6.335 decaf::internal::nio::LongArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/LongArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::LongArrayBuffer:

Public Member Functions

- **LongArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **IntArrayBuffer** (p. 1410) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **LongArrayBuffer** (long long *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **LongArrayBuffer** (p. 1666) object that wraps the given array.*

- **LongArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **LongArrayBuffer** (const **LongArrayBuffer** &other)

  *Create a **LongArrayBuffer** (p. 1666) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**LongArrayBuffer** ()
- virtual long long * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the long long array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual LongBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only long long buffer that shares this buffer's content.

- virtual LongBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual LongBuffer * **duplicate** ()

  Creates a new long long buffer that shares this buffer's content.

- virtual long long **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual long long **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible long long array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual LongBuffer & **put** (long long value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual LongBuffer & **put** (std::size_t index, long long value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual LongBuffer * **slice** () const

  Creates a new LongBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.335.1 Constructor & Destructor Documentation

6.335.1.1 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **IntArrayBuffer** (p. 1410) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.335.1.2 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (long long * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::NullPointerException)

Creates a **LongArrayBuffer** (p. 1666) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.335.1.3 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **LongArrayBuffer** (p. 1666) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the offset into array where the buffer starts

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.335.1.4 decaf::internal::nio::LongArrayBuffer::LongArrayBuffer (const LongArrayBuffer & *other*)

Create a **LongArrayBuffer** (p. 1666) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **LongArrayBuffer** (p. 1666) this one is to mirror.

6.335.1.5 virtual decaf::internal::nio::LongArrayBuffer::~LongArrayBuffer () `[virtual]`

6.335.2 Member Function Documentation

6.335.2.1 virtual long long* decaf::internal::nio::LongArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) `[virtual]`

Returns the long long array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1677).

6.335.2.2 virtual std::size_t decaf::internal::nio::LongArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::LongBuffer** (p. 1677).

6.335.2.3 virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only long long buffer which the caller then owns.

Implements **decaf::nio::LongBuffer** (p. 1677).

6.335.2.4 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this LongBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1678).

6.335.2.5 virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::duplicate () [virtual]

Creates a new long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new long long Buffer which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1678).

6.335.2.6 virtual long long decaf::internal::nio::LongArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the long long is to be read

Returns:

the long long that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::LongBuffer** (p. 1680).

6.335.2.7 virtual long long decaf::internal::nio::LongArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the long long at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::LongBuffer** (p. 1680).

6.335.2.8 virtual bool decaf::internal::nio::LongArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::LongBuffer** (p. 1680).

6.335.2.9 virtual bool decaf::internal::nio::LongArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.335.2.10 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1681).

6.335.2.11 virtual LongBuffer& decaf::internal::nio::LongArrayBuffer::put (long long *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::LongBuffer** (p. 1681).

6.335.2.12 virtual void decaf::internal::nio::LongArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.335.2.13 virtual LongBuffer* decaf::internal::nio::LongArrayBuffer::slice () const [virtual]

Creates a new LongBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create LongBuffer which the caller owns.

Implements **decaf::nio::LongBuffer** (p. 1683).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**LongArrayBuffer.h**

6.336 decaf::nio::LongBuffer Class Reference

This class defines four categories of operations upon long long buffers:.

`#include <src/main/decaf/nio/LongBuffer.h>`Inheritance diagram for decaf::nio::LongBuffer:

Public Member Functions

- virtual ~**LongBuffer** ()
- virtual std::string **toString** () const
- virtual long long * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the long long array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **LongBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only long long buffer that shares this buffer's content.

- virtual **LongBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **LongBuffer** * **duplicate** ()=0

  Creates a new long long buffer that shares this buffer's content.

- virtual long long **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual long long **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **LongBuffer** & **get** (std::vector< long long > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **LongBuffer** & **get** (long long *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible long long array.

- **LongBuffer** & **put** (**LongBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the long longs remaining in the given source buffer long longo this buffer.

- **LongBuffer** & **put** (const long long *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers long longs long longo this buffer from the given source array.

- **LongBuffer** & **put** (std::vector< long long > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source long longs array long longo this buffer.

- virtual **LongBuffer** & **put** (long long value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given long longs long longo this buffer at the current position, and then increments the position.

- virtual **LongBuffer** & **put** (std::size_t index, long long value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given long longs long longo this buffer at the given index.

- virtual **LongBuffer** * **slice** () const =0

  *Creates a new **LongBuffer** (p. 1674) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **LongBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **LongBuffer** &value) const
- virtual bool **operator==** (const **LongBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **LongBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **LongBuffer** * **allocate** (std::size_t capacity)

  Allocates a new Double buffer.

- static **LongBuffer** * **wrap** (long long *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **LongBuffer** (p. 1674).*

- static **LongBuffer** * **wrap** (std::vector< long long > &buffer)

  *Wraps the passed STL long long Vector in a **LongBuffer** (p. 1674).*

Protected Member Functions

- **LongBuffer** (std::size_t capacity)

  *Creates a **LongBuffer** (p. 1674) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted.*

6.336.1 Detailed Description

This class defines four categories of operations upon long long buffers:. o Absolute and relative get and put methods that read and write single long longs; o Relative bulk get methods that transfer contiguous sequences of long longs from this buffer long longo an array; and o Relative bulk put methods that transfer contiguous sequences of long longs from a long long array or some other long long buffer long longo this buffer o Methods for compacting, duplicating, and slicing a long long buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing long long array long longo a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.336.2 Constructor & Destructor Documentation

6.336.2.1 decaf::nio::LongBuffer::LongBuffer (std::size_t *capacity*) [protected]

Creates a **LongBuffer** (p. 1674) object that has its backing array allocated long longernally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 627) in doubles

6.336.2.2 virtual decaf::nio::LongBuffer::~LongBuffer () [inline, virtual]

6.336.3 Member Function Documentation

6.336.3.1 static LongBuffer* decaf::nio::LongBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in long longs

Returns:

the **LongBuffer** (p. 1674) that was allocated, caller owns.

6.336.3.2 virtual long long* decaf::nio::LongBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the long long array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1668).

6.336.3.3 virtual std::size_t decaf::nio::LongBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset long longo the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1669).

6.336.3.4 virtual LongBuffer* decaf::nio::LongBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only long long buffer which the caller then owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1669).

6.336.3.5 virtual LongBuffer& decaf::nio::LongBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **LongBuffer** (p. 1674)

Exceptions:

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1669).

6.336.3.6 virtual int decaf::nio::LongBuffer::compareTo (const LongBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative long longeger, zero, or a positive long longeger as this object is less than, equal to, or greater than the specified object.

6.336.3.7 virtual LongBuffer* decaf::nio::LongBuffer::duplicate () [pure virtual]

Creates a new long long buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new long long **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1670).

6.336.3.8 virtual bool decaf::nio::LongBuffer::equals (const LongBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.336.3.9 LongBuffer& decaf::nio::LongBuffer::get (long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers long longs from this buffer long longo the given destination array. If there are fewer long longs remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length long longs from this buffer long longo the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - long longer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length long longs remaining in this buffer

NullPolong longerException if the passed buffer is null.

6.336.3.10 LongBuffer& decaf::nio::LongBuffer::get (std::vector< long long > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer long longo the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length long longs remaining in this buffer

6.336.3.11 virtual long long decaf::nio::LongBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the long long is to be read

Returns:

the long long that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1670).

6.336.3.12 virtual long long decaf::nio::LongBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the long long at the current position

Exceptions:

BufferUnderflowException *(p. **661**)* if there no more data to return

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1671).

6.336.3.13 virtual bool decaf::nio::LongBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible long long array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1671).

6.336.3.14 virtual bool decaf::nio::LongBuffer::operator< (const LongBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.336.3.15 virtual bool decaf::nio::LongBuffer::operator== (const LongBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.336.3.16 virtual LongBuffer& decaf::nio::LongBuffer::put (std::size_t *index*, long long *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given long longs long longo this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the long longs to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException (p. ***2167***) - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1671).

6.336.3.17 virtual LongBuffer& decaf::nio::LongBuffer::put (long long *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given long longs long longo this buffer at the current position, and then increments the position.

Parameters:

value - the long longs value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1672).

6.336.3.18 LongBuffer& decaf::nio::LongBuffer::put (std::vector< long long > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source long longs array long longo this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **LongBuffer** (p. 1674)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.336.3.19 LongBuffer& decaf::nio::LongBuffer::put (const long long * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers long longs long longo this buffer from the given source array. If there are more long longs to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no long longs are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array long longo this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which long longs are to be read

offset- The offset within the array of the first long long to be read;

length - The number of long longs to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

NullPolong longerException if the passed buffer is null.

6.336.3.20 LongBuffer& decaf::nio::LongBuffer::put (LongBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the long longs remaining in the given source buffer long longo this buffer. If there are more long longs remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 632), then no long longs are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() long longs from the given buffer long longo this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take long longs from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. 658)* - If there is insufficient space in this buffer for the remaining long longs in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

6.336.3.21 virtual LongBuffer* decaf::nio::LongBuffer::slice () const [pure virtual]

Creates a new **LongBuffer** (p. 1674) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **LongBuffer** (p. 1674) which the caller owns.

Implemented in **decaf::internal::nio::LongArrayBuffer** (p. 1672).

6.336.3.22 virtual std::string decaf::nio::LongBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.336.3.23 static LongBuffer* decaf::nio::LongBuffer::wrap (std::vector< long long > & *buffer*) [static]

Wraps the passed STL long long Vector in a **LongBuffer** (p. 1674). The new buffer will be backed by the given long long array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **LongBuffer** (p. 1674) that is backed by buffer, caller owns.

6.336.3.24 static LongBuffer* decaf::nio::LongBuffer::wrap (long long * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **LongBuffer** (p. 1674). The new buffer will be backed by the given long long array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **LongBuffer** (p. 1674) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**LongBuffer.h**

6.337 activemq::util::LongSequenceGenerator Class Reference

This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

`#include <src/main/activemq/util/LongSequenceGenerator.h>`

Public Member Functions

- **LongSequenceGenerator** ()
- virtual ~**LongSequenceGenerator** ()
- long long **getNextSequenceId** ()
- long long **getLastSequenceId** ()

6.337.1 Detailed Description

This class is used to generate a sequence of long long values that are incremented each time a new value is requested. This class is thread safe so the ids can be requested in different **threads** (p. 66) safely.

6.337.2 Constructor & Destructor Documentation

6.337.2.1 activemq::util::LongSequenceGenerator::LongSequenceGenerator ()

6.337.2.2 virtual activemq::util::LongSequenceGenerator::~LongSequenceGenerator () [inline, virtual]

6.337.3 Member Function Documentation

6.337.3.1 long long activemq::util::LongSequenceGenerator::getLastSequenceId ()

Returns:

the last id that was generated.

6.337.3.2 long long activemq::util::LongSequenceGenerator::getNextSequenceId ()

Returns:

the next id in the sequence.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**LongSequenceGenerator.h**

6.338 decaf::net::MalformedURLException Class Reference

`#include <src/main/decaf/net/MalformedURLException.h>`Inheritance diagram for decaf::net::MalformedURLException:

Public Member Functions

- **MalformedURLException** () throw ()
  Default Constructor.
- **MalformedURLException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **MalformedURLException** (const **MalformedURLException** &ex) throw ()
  Copy Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **MalformedURLException** (const std::exception ***cause**) throw ()
  Constructor.
- **MalformedURLException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **MalformedURLException** * **clone** () const
  Clones this exception.
- virtual ~**MalformedURLException** () throw ()

6.338.1 Constructor & Destructor Documentation

6.338.1.1 decaf::net::MalformedURLException::MalformedURLException () throw () [inline]

Default Constructor.

6.338.1.2 decaf::net::MalformedURLException::MalformedURLException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.338.1.3 decaf::net::MalformedURLException::MalformedURLException (const MalformedURLException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.338.1.4 decaf::net::MalformedURLException::MalformedURLException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.338.1.5 decaf::net::MalformedURLException::MalformedURLException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.338.1.6 decaf::net::MalformedURLException::MalformedURLException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.338.1.7 virtual decaf::net::MalformedURLException::~MalformedURLException () throw () [inline, virtual]

6.338.2 Member Function Documentation

6.338.2.1 virtual MalformedURLException* decaf::net::MalformedURLException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**MalformedURLException.h**

6.339 decaf::util::Map< K, V, COMPARATOR > Class Template Reference

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

`#include <src/main/decaf/util/Map.h>`Inheritance diagram for decaf::util::Map< K, V, COMPARATOR >:

Data Structures

- class **Entry**

Public Member Functions

- **Map** ()

  Default constructor - does nothing.

- virtual ~**Map** ()
- virtual bool **equals** (const **Map** &source) const =0

  Comparison, equality is dependent on the method of determining if the element are equal.

- virtual void **copy** (const **Map** &source)=0

  Copies the content of the source map into this map.

- virtual void **clear** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  Removes all keys and values from this map.

- virtual bool **containsKey** (const K &key) const =0

  Indicates whether or this map contains a value for the given key.

- virtual bool **containsValue** (const V &value) const =0

  Indicates whether or this map contains a value for the given value, i.e.

- virtual bool **isEmpty** () const =0
- virtual std::size_t **size** () const =0
- virtual V & **get** (const K &key)=0 throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*

- virtual const V & **get** (const K &key) const =0 throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*

- virtual void **put** (const K &key, const V &value)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  Sets the value for the specified key.

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  *Stores a copy of the Mappings contained in the other **Map** (p. 1689) in this one.*

- virtual V **remove** (const K &key)=0 throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

- virtual std::vector< K > **keySet** () const =0

  *Returns a **Set** (p. 2320) view of the mappings contained in this map.*

- virtual std::vector< V > **values** () const =0

6.339.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::Map< K, V, COMPARATOR >

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

6.339.2 Constructor & Destructor Documentation

6.339.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::Map< K, V, COMPARATOR >::Map () [inline]

Default constructor - does nothing.

6.339.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::Map< K, V, COMPARATOR >::~Map () [inline, virtual]

6.339.3 Member Function Documentation

6.339.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap**< K, V, COMPARATOR > (p. 922), **decaf::util::StlMap**< K, V, COMPARATOR >

(p. 2433), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 922), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2433), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2433), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2433), decaf::util::StlMap< std::string, cms::Queue * > (p. 2433), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2433), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2433), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2433), decaf::util::StlMap< std::string, TransportFactory * > (p. 2433), decaf::util::StlMap< int, Pointer< Command > > (p. 2433), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2433), decaf::util::StlMap< std::string, CachedProducer * > (p. 2433), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2433).

6.339.3.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsKey (const K & *key*) const [pure virtual]

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 922), decaf::util::StlMap< K, V, COMPARATOR > (p. 2433), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 922), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 922), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2433), decaf::util::StlMap< std::string,

WireFormatFactory * > (p. 2433), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2433), **decaf::util::StlMap< std::string, cms::Queue** * > (p. 2433), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer** *, **commands::ProducerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, CachedConsumer** * > (p. 2433), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, TransportFactory** * > (p. 2433), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2433), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher** *, **commands::ConsumerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, CachedProducer** * > (p. 2433), and **decaf::util::StlMap< std::string, cms::Topic** * > (p. 2433).

6.339.3.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::containsValue (const V & *value*) const [pure virtual]

Indicates whether or this map contains a value for the given value, i.e. they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 923), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2433), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 923), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 923), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 923), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 923), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 923), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 923), **decaf::util::StlMap< cms::Session** *, **SessionResolver** * > (p. 2433), **decaf::util::StlMap< std::string, WireFormatFactory** * > (p. 2433), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2433), **decaf::util::StlMap< std::string, cms::Queue** * > (p. 2433), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer** *, **commands::ProducerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, CachedConsumer** * > (p. 2433), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer** *, **commands::ConsumerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, TransportFactory** * > (p. 2433), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2433), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher** *, **commands::ConsumerId::COMPARATOR >** (p. 2433), **decaf::util::StlMap< std::string, CachedProducer** * > (p. 2433), and **decaf::util::StlMap< std::string, cms::Topic** * > (p. 2433).

6.339.3.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & *source*) [pure virtual]

Copies the content of the source map into this map. Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 923), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2434).

6.339.3.5 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & *source*) const [pure virtual]

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - **Map** (p. 1689) to compare to this one.

Returns:

true if the **Map** (p. 1689) passed is equal in value to this one.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 924), and **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2434).

6.339.3.6 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::Map< K, V, COMPARATOR >::get (const K & *key*) const throw (lang::exceptions::NoSuchElementException) [pure virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689). If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 924), **decaf::util::StlMap< K, V, COMPARATOR >**

(p. 2434), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 924), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 924), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 924), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 924), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 924), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 924), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2434), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2434), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2434), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2434), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2434), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2434), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2434), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2434), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2434), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2434), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2434), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2434).

6.339.3.7 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::Map< K, V, COMPARATOR >::get (const K & *key*) throw (lang::exceptions::NoSuchElementException) [pure virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689). If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 925), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2435), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR**

> (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 925), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2435), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2435), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2435), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2435), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2435), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2435).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

6.339.3.8 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::Map< K, V, COMPARATOR >::isEmpty () const [pure virtual]

Returns:

if the **Map** (p. 1689) contains any element or not, TRUE or FALSE

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 925), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2435), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 925), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2435), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2435), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2435), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2435), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2435), and

decaf::util::StlMap< std::string, cms::Topic * > (p. 2435).

6.339.3.9 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::Map< K, V, COMPARATOR >::keySet () const [pure virtual]

Returns a **Set** (p. 2320) view of the mappings contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1490), **Set.remove** (p. 130), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns:

the entire set of keys in this map as a std::vector.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 925), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2435), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 925), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 925), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2435), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2435), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2435), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2435), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2435), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2435), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2435), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2435).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals(), and decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::equals().

6.339.3.10 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::put (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 926), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2437), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 926), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2437), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2437), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2437), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2437), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2437), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2437), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2437), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2437), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2437), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2437), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2437), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2437).

6.339.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::Map< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1689) in this one.

Parameters:

other A **Map** (p. 1689) instance whose elements are to all be inserted in this **Map** (p. 1689).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 927), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2437), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 927), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 927), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2437), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2437), and **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2437).

6.339.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::Map< K, V, COMPARATOR >::remove (const K & *key*) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 1689).

UnsupportedOperationException if this map is unmodifiable.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 929), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2438), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 929), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >**

(p. 929), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 929), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 929), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 929), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 929), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2438), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2438), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2438), decaf::util::StlMap< std::string, cms::Queue * > (p. 2438), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2438), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2438), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2438), decaf::util::StlMap< std::string, TransportFactory * > (p. 2438), decaf::util::StlMap< int, Pointer< Command > > (p. 2438), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR > (p. 2438), decaf::util::StlMap< std::string, CachedProducer * > (p. 2438), and decaf::util::StlMap< std::string, cms::Topic * > (p. 2438).

6.339.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::Map< K, V, COMPARATOR >::size () const [pure virtual]

Returns:

The number of elements (key/value pairs) in this map.

Implemented in decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR > (p. 930), decaf::util::StlMap< K, V, COMPARATOR > (p. 2438), decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR > (p. 930), decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR > (p. 930), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR > (p. 930), decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR > (p. 930), decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR > (p. 930), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 930), decaf::util::StlMap< cms::Session *, SessionResolver * > (p. 2438), decaf::util::StlMap< std::string, WireFormatFactory * > (p. 2438), decaf::util::StlMap< std::string, PrimitiveValueNode > (p. 2438), decaf::util::StlMap< std::string, cms::Queue * > (p. 2438), decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR > (p. 2438), decaf::util::StlMap< std::string, CachedConsumer * > (p. 2438), decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR > (p. 2438), decaf::util::StlMap< std::string, TransportFactory * > (p. 2438), decaf::util::StlMap< int, Pointer< Command > > (p. 2438), decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR

> (p. 2438), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2438), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2438).

6.339.3.14 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<V> decaf::util::Map< K, V, COMPARATOR >::values () const [pure virtual]

Returns:

the entire set of values in this map as a std::vector.

Implemented in **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 931), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2438), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 931), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2438), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2438), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2438), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2438), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2438), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2438), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2438), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2438), and **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2438).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Map.h**

6.340 cms::MapMessage Class Reference

A **MapMessage** (p. 1701) object is used to send a set of name-value pairs.

`#include <src/main/cms/MapMessage.h>`Inheritance diagram for cms::MapMessage:

Public Member Functions

- virtual ~**MapMessage** ()
- virtual std::vector< std::string > **getMapNames** () const =0 throw (CMSException)

  *Returns an Enumeration of all the names in the **MapMessage** (p. 1701) object.*

- virtual bool **itemExists** (const std::string &name) const =0 throw (CMSException)

  *Indicates whether an item exists in this **MapMessage** (p. 1701) object.*

- virtual bool **getBoolean** (const std::string &name) const =0 throw (CMSException)

  Returns the Boolean value of the Specified name.

- virtual void **setBoolean** (const std::string &name, bool value)=0 throw (CMSException)

  Sets a boolean value with the specified name into the Map.

- virtual unsigned char **getByte** (const std::string &name) const =0 throw (CMSException)

  Returns the Byte value of the Specified name.

- virtual void **setByte** (const std::string &name, unsigned char value)=0 throw (CMSException)

  Sets a Byte value with the specified name into the Map.

- virtual std::vector< unsigned char > **getBytes** (const std::string &name) const =0 throw (CMSException)

  Returns the Bytes value of the Specified name.

- virtual void **setBytes** (const std::string &name, const std::vector< unsigned char > &value)=0 throw (CMSException)

  Sets a Bytes value with the specified name into the Map.

- virtual char **getChar** (const std::string &name) const =0 throw (CMSException)

  Returns the Char value of the Specified name.

- virtual void **setChar** (const std::string &name, char value)=0 throw (CMSException)

  Sets a Char value with the specified name into the Map.

- virtual double **getDouble** (const std::string &name) const =0 throw (CMSException)

  Returns the Double value of the Specified name.

- virtual void **setDouble** (const std::string &name, double value)=0 throw (CMSException)

Sets a Double value with the specified name into the Map.

- virtual float **getFloat** (const std::string &name) const =0 throw (CMSException)
  Returns the Float value of the Specified name.
- virtual void **setFloat** (const std::string &name, float value)=0 throw (CMSException)
  Sets a Float value with the specified name into the Map.
- virtual int **getInt** (const std::string &name) const =0 throw (CMSException)
  Returns the Int value of the Specified name.
- virtual void **setInt** (const std::string &name, int value)=0 throw (CMSException)
  Sets a Int value with the specified name into the Map.
- virtual long long **getLong** (const std::string &name) const =0 throw (CMSException)
  Returns the Long value of the Specified name.
- virtual void **setLong** (const std::string &name, long long value)=0 throw (CMSException)
  Sets a Long value with the specified name into the Map.
- virtual short **getShort** (const std::string &name) const =0 throw (CMSException)
  Returns the Short value of the Specified name.
- virtual void **setShort** (const std::string &name, short value)=0 throw (CMSException)
  Sets a Short value with the specified name into the Map.
- virtual std::string **getString** (const std::string &name) const =0 throw (CMSException)
  Returns the String value of the Specified name.
- virtual void **setString** (const std::string &name, const std::string &value)=0 throw (CMSException)
  Sets a String value with the specified name into the Map.

6.340.1 Detailed Description

A **MapMessage** (p. 1701) object is used to send a set of name-value pairs. The names are String objects, and the values are primitive data types in the Java programming language. The names must have a value that is not null, and not an empty string. The entries can be accessed sequentially or randomly by name. The order of the entries is undefined. **MapMessage** (p. 1701) inherits from the **Message** (p. 1753) interface and adds a message body that contains a Map.

When a client receives a **MapMessage** (p. 1701), it is in read-only mode. If a client attempts to write to the message at this point, a **MessageNotWriteableException** (p. 1877) is thrown. To place the **MapMessage** (p. 1701) back into a state where it can be read from and written to, call the `clearBody` method.

MapMessage (p. 1701) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 850). The String-to-primitive conversions may throw a **MessageFormatException** (p. 1842) if the primitive's valueOf() method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X								X	
byte		X	X		X	X			X	
short			X		X	X			X	
char				X					X	
int					X	X			X	
long						X			X	
float							X	X	X	
double								X	X	
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.0

6.340.2 Constructor & Destructor Documentation

6.340.2.1 virtual cms::MapMessage::~MapMessage () [inline, virtual]

6.340.3 Member Function Documentation

6.340.3.1 virtual bool cms::MapMessage::getBoolean (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Boolean value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(*p. ***850****)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 259).

6.340.3.2 virtual unsigned char cms::MapMessage::getByte (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Byte value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(*p. ***850****)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 259).

6.340.3.3 virtual std::vector<unsigned char> cms::MapMessage::getBytes (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Bytes value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 259).

6.340.3.4 virtual char cms::MapMessage::getChar (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Char value of the Specified name.

Parameters:

name name of the value to fetch from the map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 260).

6.340.3.5 virtual double cms::MapMessage::getDouble (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Double value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 260).

6.340.3.6 virtual float cms::MapMessage::getFloat (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Float value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(*p. ***850****)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 260).

6.340.3.7 virtual int cms::MapMessage::getInt (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Int value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(*p. ***850****)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 261).

6.340.3.8 virtual long long cms::MapMessage::getLong (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Long value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(*p. ***850****)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 261).

6.340.3.9 virtual std::vector< std::string > cms::MapMessage::getMapNames () const throw (CMSException) [pure virtual]

Returns an Enumeration of all the names in the **MapMessage** (p. 1701) object.

Returns:

STL Vector of String values, each of which is the name of an item in the **MapMessage** (p. 1701)

Exceptions:

CMSException *(p. **850**)* - if the operation fails due to an internal error.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 262).

6.340.3.10 virtual short cms::MapMessage::getShort (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the Short value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(p. **850**)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 262).

6.340.3.11 virtual std::string cms::MapMessage::getString (const std::string & *name*) const throw (CMSException) [pure virtual]

Returns the String value of the Specified name.

Parameters:

name Name of the value to fetch from the map

Exceptions:

CMSException *(p. **850**)* - if the operation fails due to an internal error.

MessageFormatExceptio - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 262).

6.340.3.12 virtual bool cms::MapMessage::itemExists (const std::string & *name*) const throw (CMSException) [pure virtual]

Indicates whether an item exists in this **MapMessage** (p. 1701) object.

Parameters:

name String name of the Object in question

Returns:

boolean value indicating if the name is in the map

Exceptions:

CMSException *(*p. ***850)*** - if the operation fails due to an internal error.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 263).

6.340.3.13 virtual void cms::MapMessage::setBoolean (const std::string & *name*, bool *value*) throw (CMSException) [pure virtual]

Sets a boolean value with the specified name into the Map.

Parameters:

name the name of the boolean

value the boolean value to set in the Map

Exceptions:

CMSException *(*p. ***850)*** - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 263).

6.340.3.14 virtual void cms::MapMessage::setByte (const std::string & *name*, unsigned char *value*) throw (CMSException) [pure virtual]

Sets a Byte value with the specified name into the Map.

Parameters:

name the name of the Byte

value the Byte value to set in the Map

Exceptions:

CMSException *(*p. ***850)*** - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 263).

6.340.3.15 virtual void cms::MapMessage::setBytes (const std::string & *name*, const std::vector< unsigned char > & *value*) throw (CMSException) [pure virtual]

Sets a Bytes value with the specified name into the Map.

Parameters:

name The name of the Bytes

value The Bytes value to set in the Map

Exceptions:

CMSException *(*p. ***850)*** - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 264).

6.340.3.16 virtual void cms::MapMessage::setChar (const std::string & *name*, char *value*) throw (CMSException) [pure virtual]

Sets a Char value with the specified name into the Map.

Parameters:

name the name of the Char

value the Char value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 264).

6.340.3.17 virtual void cms::MapMessage::setDouble (const std::string & *name*, double *value*) throw (CMSException) [pure virtual]

Sets a Double value with the specified name into the Map.

Parameters:

name The name of the Double

value The Double value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 264).

6.340.3.18 virtual void cms::MapMessage::setFloat (const std::string & *name*, float *value*) throw (CMSException) [pure virtual]

Sets a Float value with the specified name into the Map.

Parameters:

name The name of the Float

value The Float value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 265).

6.340.3.19 virtual void cms::MapMessage::setInt (const std::string & *name*, int *value*) throw (CMSException) [pure virtual]

Sets a Int value with the specified name into the Map.

Parameters:

name The name of the Int

value The Int value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 265).

6.340.3.20 virtual void cms::MapMessage::setLong (const std::string & *name*, long long *value*) throw (CMSException) [pure virtual]

Sets a Long value with the specified name into the Map.

Parameters:

name The name of the Long

value The Long value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 265).

6.340.3.21 virtual void cms::MapMessage::setShort (const std::string & *name*, short *value*) throw (CMSException) [pure virtual]

Sets a Short value with the specified name into the Map.

Parameters:

name The name of the Short

value The Short value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 266).

6.340.3.22 virtual void cms::MapMessage::setString (const std::string & *name*, const std::string & *value*) throw (CMSException) [pure virtual]

Sets a String value with the specified name into the Map.

Parameters:

name The name of the String

value The String value to set in the Map

Exceptions:

CMSException (p. ***850***) - if the operation fails due to an internal error.

MessageNotWritableException - if the **Message** (p. 1753) is in Read-only Mode.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 266).

The documentation for this class was generated from the following file:

- src/main/cms/**MapMessage.h**

6.341 decaf::util::logging::MarkBlockLogger Class Reference

Defines a class that can be used to mark the entry and exit from scoped blocks.

`#include <src/main/decaf/util/logging/MarkBlockLogger.h>`

Public Member Functions

- **MarkBlockLogger** (**Logger** *logger, const std::string &blockName)
  Constructor - Marks Block entry.
- virtual ~**MarkBlockLogger** ()

6.341.1 Detailed Description

Defines a class that can be used to mark the entry and exit from scoped blocks. Create an instance of this class at the start of a scoped block, passing it the logger to use and the name of the block. The block entry and exit will be marked using the scope name, logger to the logger at the MARKBLOCK log level.

6.341.2 Constructor & Destructor Documentation

6.341.2.1 decaf::util::logging::MarkBlockLogger::MarkBlockLogger (Logger * *logger*, const std::string & *blockName*) [inline]

Constructor - Marks Block entry.

Parameters:

logger **Logger** (p. 1623) to use

blockName Block name

6.341.2.2 virtual decaf::util::logging::MarkBlockLogger::~MarkBlockLogger () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**MarkBlockLogger.h**

6.342 activemq::wireformat::MarshalAware Class Reference

`#include <src/main/activemq/wireformat/MarshalAware.h>`Inheritance diagram for activemq::wireformat::MarshalAware:

Public Member Functions

- virtual ~**MarshalAware** ()
- virtual bool **isMarshalAware** () const =0

  Determine if the class implementing this interface is really wanting to be told about marshaling.

- virtual void **beforeMarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)

  Called before marshaling is started to prepare the object to be marshaled.

- virtual void **afterMarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)

  Called after marshaling is started to cleanup the object being marshaled.

- virtual void **beforeUnmarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)

  Called before unmarshaling is started to prepare the object to be unmarshaled.

- virtual void **afterUnmarshal** (**WireFormat** *wireFormat)=0 throw (decaf::io::IOException)

  Called after unmarshaling is started to cleanup the object being unmarshaled.

- virtual void **setMarshaledForm** (**WireFormat** *wireFormat, const std::vector< char > &data)=0

  Called to set the data to this object that will contain the objects marshaled form.

- virtual std::vector< unsigned char > **getMarshaledForm** (**WireFormat** *wireFormat)=0

  Called to get the data to this object that will contain the objects marshaled form.

6.342.1 Constructor & Destructor Documentation

6.342.1.1 virtual activemq::wireformat::MarshalAware::~MarshalAware () [inline, virtual]

6.342.2 Member Function Documentation

6.342.2.1 virtual void activemq::wireformat::MarshalAware::afterMarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called after marshaling is started to cleanup the object being marshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control marshaling

6.342.2.2 virtual void activemq::wireformat::MarshalAware::afterUnmarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

6.342.2.3 virtual void activemq::wireformat::MarshalAware::beforeMarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called before marshaling is started to prepare the object to be marshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control marshaling

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 257), and **activemq::commands::ActiveMQStreamMessage** (p. 376).

6.342.2.4 virtual void activemq::wireformat::MarshalAware::beforeUnmarshal (WireFormat * *wireFormat*) throw (decaf::io::IOException) [pure virtual]

Called before unmarshaling is started to prepare the object to be unmarshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

6.342.2.5 virtual std::vector<unsigned char> activemq::wireformat::MarshalAware::getMarshaledForm (WireFormat * *wireFormat*) [pure virtual]

Called to get the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

Returns:

buffer that holds the objects data.

6.342.2.6 virtual bool activemq::wireformat::MarshalAware::isMarshalAware () const [pure virtual]

Determine if the class implementing this interface is really wanting to be told about marshaling. Normally if you didn't want to be marshal aware you just wouldn't implement this interface but since this is C++ and we don't have true interfaces we need a flat inheritance hierarchy, so we always implement this.

Returns:

true if this class cares about marshaling.

Implemented in **activemq::commands::ActiveMQMapMessage** (p. 262), **activemq::commands::ActiveMQStreamMessage** (p. 378), **activemq::commands::BaseDataStructure** (p. 559), **activemq::commands::Message** (p. 1746), and **activemq::commands::WireFormatInfo** (p. 2704).

6.342.2.7 virtual void activemq::wireformat::MarshalAware::setMarshaledForm (WireFormat * *wireFormat*, const std::vector< char > & *data*) [pure virtual]

Called to set the data to this object that will contain the objects marshaled form.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

data - vector of object binary data

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**MarshalAware.h**

6.343 activemq::wireformat::openwire::marshal::v2::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MarshallerFactory.h>`

Public Member Functions

- virtual ~**MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** *format)

6.343.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.343.2 Constructor & Destructor Documentation

6.343.2.1 virtual activemq::wireformat::openwire::marshal::v2::MarshallerFactory::~MarshallerFactory () `[inline, virtual]`

6.343.3 Member Function Documentation

6.343.3.1 virtual void activemq::wireformat::openwire::marshal::v2::MarshallerFactory::configure (OpenWireFormat * *format*) `[virtual]`

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MarshallerFactory.h**

6.344 activemq::wireformat::openwire::marshal::v3::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h>`

Public Member Functions

- virtual ~**MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** *format)

6.344.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.344.2 Constructor & Destructor Documentation

6.344.2.1 virtual activemq::wireformat::openwire::marshal::v3::MarshallerFactory::~MarshallerFactory () `[inline, virtual]`

6.344.3 Member Function Documentation

6.344.3.1 virtual void activemq::wireformat::openwire::marshal::v3::MarshallerFactory::configure (OpenWireFormat * *format*) `[virtual]`

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MarshallerFactory.h**

6.345 activemq::wireformat::openwire::marshal::v1::MarshallerFactory Class Reference

Used to create marshallers for a specific version of the wire protocol.

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h>`

Public Member Functions

- virtual ~**MarshallerFactory** ()
- virtual void **configure** (**OpenWireFormat** *format)

6.345.1 Detailed Description

Used to create marshallers for a specific version of the wire protocol. NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Groovy scripts in the activemq-openwire-generator module

6.345.2 Constructor & Destructor Documentation

6.345.2.1 virtual activemq::wireformat::openwire::marshal::v1::MarshallerFactory::~MarshallerFactory () `[inline, virtual]`

6.345.3 Member Function Documentation

6.345.3.1 virtual void activemq::wireformat::openwire::marshal::v1::MarshallerFactory::configure (OpenWireFormat * *format*) `[virtual]`

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MarshallerFactory.h**

6.346 decaf::lang::Math Class Reference

The class `Math` (p. 1718) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

`#include <src/main/decaf/lang/Math.h>`

Public Member Functions

- **Math** ()
- virtual ~**Math** ()

Static Public Member Functions

- static int **abs** (int value)

  Returns the absolute value of an int value.

- static long long **abs** (long long value)

  Returns the absolute value of an long long value.

- static float **abs** (float value)

  Returns the absolute value of a float value.

- static double **abs** (double value)

  Returns the absolute value of a double value.

- static double **sqrt** (double value)

  Returns the arc cosine of an angle, in the range of 0.0 through pi.

- static double **pow** (double base, double exp)

  Returns the value of the first argument raised to the power of the second argument.

- static short **min** (short a, short b)

  Returns the double value that is closest in value to the argument and is equal to a mathematical integer.

- static int **min** (int a, int b)

  Returns the smaller of two ***int*** *values.*

- static int **min** (unsigned int a, unsigned int b)

  Returns the smaller of two ***unsigned int*** *values.*

- static long long **min** (long long a, long long b)

  Returns the smaller of two ***long long*** *values.*

- static float **min** (float a, float b)

  Returns the smaller of two float values.

- static double **min** (double a, double b)

  Returns the smaller of two double values.

- static short **max** (short a, short b)

  Returns the larger of two `short` *values.*

- static int **max** (int a, int b)

  Returns the larger of two `int` *values.*

- static long long **max** (long long a, long long b)

  Returns the larger of two `long long` *values.*

- static float **max** (float a, float b)

  Returns the greater of two float values.

- static double **max** (double a, double b)

  Returns the greater of two double values.

- static double **ceil** (double value)

  Returns the natural logarithm (base e) of a double value.

- static double **floor** (double value)

  Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer.

- static int **round** (float value)

  Returns the closest int to the argument.

- static long long **round** (double value)

  Returns the closest long long to the argument.

- static double **random** ()

  Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard.

- static float **signum** (float value)

  Returns Euler's number e raised to the power of a double value.

- static double **signum** (double value)

  Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero.

- static double **toRadians** (double angdeg)

  Returns the measure in radians of the supplied degree angle.

- static double **toDegrees** (double angrad)

  Returns the measure in degrees of the supplied radian angle.

Static Public Attributes

- static const double **E**
- static const double **PI**

6.346.1 Detailed Description

The class `Math` (p. 1718) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.

6.346.2 Constructor & Destructor Documentation

6.346.2.1 decaf::lang::Math::Math () [inline]

6.346.2.2 virtual decaf::lang::Math::~Math () [inline, virtual]

6.346.3 Member Function Documentation

6.346.3.1 static double decaf::lang::Math::abs (double *value*) [static]

Returns the absolute value of a double value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Double::longBitsToDouble** (p. 1237)(0x7fffffffffffffffULL & Double::doubleToLongBits(value))

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.346.3.2 static float decaf::lang::Math::abs (float *value*) [static]

Returns the absolute value of a float value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned. Special cases:

o If the argument is positive zero or negative zero, the result is positive zero. o If the argument is infinite, the result is positive infinity. o If the argument is NaN, the result is NaN.

In other words, the result is the same as the value of the expression: **Float::intBitsToFloat** (p. 1333)(0x7fffffff & Float::floatToIntBits(value))

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.346.3.3 static long long decaf::lang::Math::abs (long long *value*) [inline, static]

Returns the absolute value of an long long value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.346.3.4 static int decaf::lang::Math::abs (int *value*) [inline, static]

Returns the absolute value of an int value. If the argument is not negative, the argument is returned. If the argument is negative, the negation of the argument is returned.

Parameters:

value - the value to return the abs of

Returns:

the value if positive, otherwise the negative of value

6.346.3.5 static double decaf::lang::Math::ceil (double *value*) [static]

Returns the natural logarithm (base e) of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity.

Parameters:

value the value to compute the natural log of.

Returns:

the natural log of value. Returns the base 10 logarithm of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is negative infinity. o If the argument is equal to 10n for integer n, then the result is n.

Parameters:

value - the value to operate on

Returns:

the long base 10 of value Returns the natural logarithm of the sum of the argument and 1. Note that for small values x, the result of log1p(x) is much closer to the true result of ln(1 + x) than the floating-point evaluation of log(1.0+x).

Special cases:

o If the argument is NaN or less than -1, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative one, then the result is negative infinity. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to operate on

Returns:

the the value ln(x + 1), the natural log of x + 1 Returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. o If the argument value is less than zero but greater than -1.0, then the result is negative zero.

Note that the value of Math.ceil(x) is exactly the value of -Math.floor(-x).

Parameters:

value - the value to find the ceiling of

Returns:

the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer.

6.346.3.6 static double decaf::lang::Math::floor (double *value*) [static]

Returns the largest (closest to positive infinity) double value that is less than or equal to the argument and is equal to a mathematical integer. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the value to find the floor of

Returns:

the largest (closest to positive infinity) floating-point value that less than or equal to the argument and is equal to a mathematical integer.

6.346.3.7 static double decaf::lang::Math::max (double *a*, double *b*) [static]

Returns the greater of two double values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of `a` and `b`.

6.346.3.8 static float decaf::lang::Math::max (float *a*, float *b*) [static]

Returns the greater of two float values. That is, the result is the argument closer to positive infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other negative zero, the result is positive zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of `a` and `b`.

6.346.3.9 static long long decaf::lang::Math::max (long long *a*, long long *b*) [inline, static]

Returns the larger of two `long long` values. That is, the result the argument closer to the value of `Long::MAX_VALUE` (p. 1665). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of `a` and `b`.

6.346.3.10 static int decaf::lang::Math::max (int *a*, int *b*) [inline, static]

Returns the larger of two `int` values. That is, the result the argument closer to the value of `Integer::MAX_VALUE` (p. 1441). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of `a` and `b`.

6.346.3.11 static short decaf::lang::Math::max (short *a*, short *b*) [inline, static]

Returns the larger of two `short` values. That is, the result the argument closer to the value of `Short::MAX_VALUE` (p. 2329). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the larger of `a` and `b`.

6.346.3.12 static double decaf::lang::Math::min (double *a*, double *b*) [static]

Returns the smaller of two double values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.13 static float decaf::lang::Math::min (float *a*, float *b*) [static]

Returns the smaller of two float values. That is, the result is the value closer to negative infinity. If the arguments have the same value, the result is that same value. If either value is NaN, then the result is NaN. Unlike the numerical comparison operators, this method considers negative zero to be strictly smaller than positive zero. If one argument is positive zero and the other is negative zero, the result is negative zero.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.14 static long long decaf::lang::Math::min (long long *a*, long long *b*) [inline, static]

Returns the smaller of two `long long` values. That is, the result the argument closer to the value of `Long::MIN_VALUE` (p. 1665). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.15 static int decaf::lang::Math::min (unsigned int *a*, unsigned int *b*) [inline, static]

Returns the smaller of two `unsigned int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 1441). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.16 static int decaf::lang::Math::min (int *a*, int *b*) [inline, static]

Returns the smaller of two `int` values. That is, the result the argument closer to the value of `Integer::MIN_VALUE` (p. 1441). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.17 static short decaf::lang::Math::min (short *a*, short *b*) [inline, static]

Returns the double value that is closest in value to the argument and is equal to a mathematical integer. If two double values that are mathematical integers are equally close, the result is the integer value that is even. Special cases:

o If the argument value is already equal to a mathematical integer, then the result is the same as the argument. o If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the value to round to the nearest integer

Returns:

the rounded value Returns the smaller of two short values. That is, the result is the argument closer to the value of `decaf.lang.Short::MIN_VALUE` (p. 2329). If the arguments have the same value, the result is that same value.

Parameters:

a - an argument.

b - another argument.

Returns:

the smaller of `a` and `b`.

6.346.3.18 static double decaf::lang::Math::pow (double *base*, double *exp*) [static]

Returns the value of the first argument raised to the power of the second argument. Special cases:

o If the second argument is positive or negative zero, then the result is 1.0. o If the second argument is 1.0, then the result is the same as the first argument. o If the second argument is NaN, then the result is NaN. o If the first argument is NaN and the second argument is nonzero, then the result is NaN.

Parameters:

base - the base

exp - the exponent

Returns:

the base raised to the power of exp.

6.346.3.19 static double decaf::lang::Math::random () [static]

Computes the remainder operation on two arguments as prescribed by the IEEE 754 standard. The remainder value is mathematically equal to f1 - f2 $\times$ n, where n is the mathematical integer closest to the exact mathematical value of the quotient f1/f2, and if two mathematical integers are equally close to f1/f2, then n is the integer that is even. If the remainder is zero, its sign is the same as the sign of the first argument. Special cases:

o If either argument is NaN, or the first argument is infinite, or the second argument is positive zero or negative zero, then the result is NaN. o If the first argument is finite and the second argument is infinite, then the result is the same as the first argument.

Parameters:

f1 - the dividend.

f2 - the divisor

Returns:

the IEEE remainder of value Returns a double value with a positive sign, greater than or equal to 0.0 and less than 1.0. Returned values are chosen pseudorandomly with (approximately) uniform distribution from that range.

When this method is first called, it creates a single new pseudorandom-number generator; This new pseudorandom-number generator is used thereafter for all calls to this method and is used nowhere else.

This method is properly synchronized to allow correct use by more than one thread. However, if many threads need to generate pseudorandom numbers at a great rate, it may reduce contention for each thread to have its own pseudorandom-number generator.

Returns:

a pseudorandom double greater than or equal to 0.0 and less than 1.0.

6.346.3.20 static long long decaf::lang::Math::round (double *value*) [static]

Returns the closest long long to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type long long. In other words, the result is equal to the value of the expression: (long long)**Math.floor** (p. 1722)(a + 0.5d)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Long::MIN_VALUE** (p. 1665), the result is equal to the value of **Long::MIN_VALUE** (p. 1665). o If the argument is positive infinity or any value greater than or equal to the value of **Long::MAX_VALUE** (p. 1665), the result is equal to the value of **Long::MAX_VALUE** (p. 1665).

Parameters:

value - the value to round

Returns:

the value of the argument rounded to the nearest integral value.

6.346.3.21 static int decaf::lang::Math::round (float *value*) [static]

Returns the closest int to the argument. The result is rounded to an integer by adding 1/2, taking the floor of the result, and casting the result to type int. In other words, the result is equal to the value of the expression: (int)**Math.floor** (p. 1722)(a + 0.5f)

o If the argument is NaN, the result is 0. o If the argument is negative infinity or any value less than or equal to the value of **Integer::MIN_VALUE** (p. 1441), the result is equal to the value of **Integer::MIN_VALUE** (p. 1441). o If the argument is positive infinity or any value greater than or equal to the value of **Integer::MAX_VALUE** (p. 1441), the result is equal to the value of **Integer::MAX_VALUE** (p. 1441).

Parameters:

value - the value to round

Returns:

the value of the argument rounded to the nearest integral value.

6.346.3.22 static double decaf::lang::Math::signum (double *value*) [static]

Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the floating-point value whose signum is to be returned

Returns:

the signum function of the argument

6.346.3.23 static float decaf::lang::Math::signum (float *value*) [static]

Returns Euler's number e raised to the power of a double value. Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is positive zero.

Parameters:

value - the exponent to raise e to

Returns:

the value e^a, where e is the base of the natural logarithms. Returns e^x - 1. Note that for values of x near 0, the exact sum of expm1(x) + 1 is much closer to the true result of ex than exp(x). Special cases:

o If the argument is NaN, the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is negative infinity, then the result is -1.0. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to raise e^x - 1

Returns:

the value ex - 1. Returns sqrt(x^2 + y^2) without intermediate overflow or underflow. Special cases:

If either argument is infinite, then the result is positive infinity. If either argument is NaN and neither argument is infinite, then the result is NaN.

Parameters:

x - an argument

y - another argument

Returns:

the sqrt(x^2 + y^2) without intermediate overflow or underflow Returns the signum function of the argument; zero if the argument is zero, 1.0f if the argument is greater than zero, -1.0f if the argument is less than zero. Special Cases:

o If the argument is NaN, then the result is NaN. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Parameters:

value - the floating-point value whose signum is to be returned

Returns:

the signum function of the argument

6.346.3.24 static double decaf::lang::Math::sqrt (double *value*) [static]

Returns the arc cosine of an angle, in the range of 0.0 through pi. Special case:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc cosine of value in radians. Returns the arc sine of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN or its absolute value is greater than 1, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc cosine of value in radians. Returns the arc tangent of an angle, in the range of -pi/2 through pi/2. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the value to return the arc cosine of.

Returns:

arc tangent of value in radians. Converts rectangular coordinates (x, y) to polar (r, theta). This method computes the phase theta by computing an arc tangent of y/x in the range of -pi to pi. Special cases:

o If either argument is NaN, then the result is NaN. o If the first argument is positive zero and the second argument is positive, or the first argument is positive and finite and the second argument is positive infinity, then the result is positive zero. o If the first argument is negative zero and the second argument is positive, or the first argument is negative and finite and the second argument is positive infinity, then the result is negative zero. o If the first argument is positive zero and the second argument is negative, or the first argument is positive and finite and the second argument is negative infinity, then the result is the double value closest to pi. o If the first argument is negative zero and the second argument is negative, or the first argument is negative and finite and the second argument is negative infinity, then the result is the double value closest to -pi. o If the first argument is positive and the second argument is positive zero or negative zero, or the first argument is positive infinity and the second argument is finite, then the result is the double value closest to pi/2. o If the first argument is negative and the second argument is positive zero or negative zero, or the first argument is negative infinity and the second argument is finite, then the result is the double value closest to -pi/2. o If both arguments are positive infinity, then the result is the double value closest to pi/4. o If the first argument is positive infinity and the second argument is negative infinity, then the result is the double value closest to 3∗pi/4. o If the first argument is negative infinity and the second argument is positive infinity, then the result is the double value closest to -pi/4. o If both arguments are negative infinity, then the result is the double value closest to -3∗pi/4.

Parameters:

y - the ordinate coordinate

x - the abscissa coordinate

Returns:

the theta component of the point (r, theta) in polar coordinates that corresponds to the point (x, y) in Cartesian coordinates. Returns the cube root of a double value. For positive finite x, cbrt(-x) == -cbrt(x); that is, the cube root of a negative value is the negative of the cube root of that value's magnitude. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the double to compute the cube root of

Returns:

the cube root of value Returns the trigonometric cosine of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN.

Parameters:

value - an value in radians

Returns:

the cosine of the argument. Returns the hyperbolic cosine of a double value. The hyperbolic cosine of x is defined to be (ex + e-x)/2 where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is positive infinity. o If the argument is zero, then the result is 1.0.

Parameters:

value - the number whose hyperbolic cosine is to be found

Returns:

the hyperbolic cosine of value Returns the trigonometric sine of an angle. Special case:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose sin is to be found

Returns:

the sine of value Returns the hyperbolic sine of a double value. The hyperbolic sine of x is defined to be (ex - e-x)/2 where e is Euler's number. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is infinite, then the result is an infinity with the same sign as the argument. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose hyperbolic sin is to be found

Returns:

the hyperbolic sine of value Returns the trigonometric tangent of an angle. Special cases:

o If the argument is NaN or an infinity, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument.

Parameters:

value - the number whose tangent is to be found

Returns:

the tangent of value Returns the hyperbolic tangent of a double value. The hyperbolic tangent of x is defined to be (ex - e-x)/(ex + e-x), in other words, sinh(x)/cosh(x). Note that the absolute value of the exact tanh is always less than 1. Special cases:

o If the argument is NaN, then the result is NaN. o If the argument is zero, then the result is a zero with the same sign as the argument. o If the argument is positive infinity, then the result is +1.0. o If the argument is negative infinity, then the result is -1.0.

Parameters:

value - the number whose hyperbolic tangent is to be found

Returns:

the hyperbolic cosine of value Returns the correctly rounded positive square root of a double value. Special cases:

o If the argument is NaN or less than zero, then the result is NaN. o If the argument is positive infinity, then the result is positive infinity. o If the argument is positive zero or negative zero, then the result is the same as the argument.

Otherwise, the result is the double value closest to the true mathematical square root of the argument value.

Parameters:

value - the value to find the square root of

the square root of the argument.

6.346.3.25 static double decaf::lang::Math::toDegrees (double *angrad*) [inline, static]

Returns the measure in degrees of the supplied radian angle.

Parameters:

angrad - an angle in radians

Returns:

the degree measure of the angle.

6.346.3.26 static double decaf::lang::Math::toRadians (double *angdeg*) [inline, static]

Returns the measure in radians of the supplied degree angle.

Parameters:

angdeg - an angle in degrees

Returns:

the radian measure of the angle.

6.346.4 Field Documentation

6.346.4.1 const double decaf::lang::Math::E [static]

6.346.4.2 const double decaf::lang::Math::PI [static]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Math.h**

6.347 activemq::util::MemoryUsage Class Reference

`#include <src/main/activemq/util/MemoryUsage.h>`Inheritance diagram for activemq::util::MemoryUsage:

Public Member Functions

- **MemoryUsage** ()

  Default Constructor.

- **MemoryUsage** (unsigned long long limit)

  *Creates an instance of an **Usage** (p. 2681) monitor with a set limit.*

- virtual ~**MemoryUsage** ()
- virtual void **waitForSpace** ()

  *Waits forever for more space to be returned to this **Usage** (p. 2681) Manager.*

- virtual void **waitForSpace** (unsigned int timeout)

  *Waits for more space to be returned to this **Usage** (p. 2681) Manager, times out when the given time span in milliseconds elapses.*

- virtual void **enqueueUsage** (unsigned long long value)

  Tries to increase the usage by value amount but blocks if this object is currently full.

- virtual void **increaseUsage** (unsigned long long value)

  Increases the usage by the value amount.

- virtual void **decreaseUsage** (unsigned long long value)

  Decreases the usage by the value amount.

- virtual bool **isFull** () const

  *Returns true if this **Usage** (p. 2681) instance is full, i.e.*

- unsigned long long **getUsage** () const

  Gets the current usage amount.

- void **setUsage** (unsigned long long usage)

  Sets the current usage amount.

- unsigned long long **getLimit** () const

  Gets the current limit amount.

- void **setLimit** (unsigned long long limit)

  Sets the current limit amount.

6.347.1 Constructor & Destructor Documentation

6.347.1.1 activemq::util::MemoryUsage::MemoryUsage ()

Default Constructor.

6.347.1.2 activemq::util::MemoryUsage::MemoryUsage (unsigned long long *limit*)

Creates an instance of an **Usage** (p. 2681) monitor with a set limit.

Parameters:

limit - amount of memory this manager allows.

6.347.1.3 virtual activemq::util::MemoryUsage::~MemoryUsage () [virtual]

6.347.2 Member Function Documentation

6.347.2.1 virtual void activemq::util::MemoryUsage::decreaseUsage (unsigned long long *value*) [virtual]

Decreases the usage by the value amount.

Parameters:

value Amount of space to return to the pool

Implements **activemq::util::Usage** (p. 2681).

6.347.2.2 virtual void activemq::util::MemoryUsage::enqueueUsage (unsigned long long *value*) [inline, virtual]

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters:

value Amount of usage in bytes to add.

Implements **activemq::util::Usage** (p. 2681).

6.347.2.3 unsigned long long activemq::util::MemoryUsage::getLimit () const [inline]

Gets the current limit amount.

Returns:

the amount that can be used before full.

6.347.2.4 unsigned long long activemq::util::MemoryUsage::getUsage () const [inline]

Gets the current usage amount.

Returns:

the amount of bytes currently used.

6.347.2.5 virtual void activemq::util::MemoryUsage::increaseUsage (unsigned long long *value*) [virtual]

Increases the usage by the value amount.

Parameters:

value Amount of usage to add.

Implements **activemq::util::Usage** (p. 2682).

6.347.2.6 virtual bool activemq::util::MemoryUsage::isFull () const [virtual]

Returns true if this **Usage** (p. 2681) instance is full, i.e. **Usage** (p. 2681) >= 100%

Implements **activemq::util::Usage** (p. 2682).

6.347.2.7 void activemq::util::MemoryUsage::setLimit (unsigned long long *limit*) [inline]

Sets the current limit amount.

Parameters:

limit - The amount that can be used before full.

6.347.2.8 void activemq::util::MemoryUsage::setUsage (unsigned long long *usage*) [inline]

Sets the current usage amount.

Parameters:

usage - The amount to tag as used.

6.347.2.9 virtual void activemq::util::MemoryUsage::waitForSpace (unsigned int *timeout*) [virtual]

Waits for more space to be returned to this **Usage** (p. 2681) Manager, times out when the given time span in milliseconds elapses.

Parameters:

timeout The time to wait for more space.

Implements **activemq::util::Usage** (p. 2682).

6.347.2.10 virtual void activemq::util::MemoryUsage::waitForSpace () [virtual]

Waits forever for more space to be returned to this **Usage** (p. 2681) Manager.

Implements **activemq::util::Usage** (p. 2682).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**MemoryUsage.h**

6.348 activemq::commands::Message Class Reference

`#include <src/main/activemq/commands/Message.h>`Inheritance diagram for activemq::commands::Message:

Public Member Functions

- **Message** ()
- virtual ~**Message** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **Message** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Called after unmarshaling is started to cleanup the object being unmarshaled.

- virtual bool **isMarshalAware** () const

  Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

- virtual void **setAckHandler** (**core::ActiveMQAckHandler** *handler)

  *Sets the Acknowledgment Handler that this **Message** (p. 1737) will use when the Acknowledge method is called.*

- virtual **core::ActiveMQAckHandler** * **getAckHandler** () const

  *Gets the Acknowledgment Handler that this **Message** (p. 1737) will use when the Acknowledge method is called.*

- virtual unsigned int **getSize** () const

  Returns the Size of this message in Bytes.

- virtual bool **isExpired** () const

  Returns if this message has expired, meaning that its Expiration time has elapsed.

- **util::PrimitiveMap** & **getMessageProperties** ()

  Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

- const **util::PrimitiveMap** & **getMessageProperties** () const
- bool **isReadOnlyProperties** () const

  *Returns if the **Message** (p. 1737) Properties Are Read Only.*

- void **setReadOnlyProperties** (bool value)

  *Set the Read Only State of the **Message** (p. 1737) Properties.*

- bool **isReadOnlyBody** () const

  *Returns if the **Message** (p. 1737) Body is Read Only.*

- void **setReadOnlyBody** (bool value)

  *Set the Read Only State of the **Message** (p. 1737) Content.*

- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getOriginalDestination** ()
- virtual void **setOriginalDestination** (const **Pointer**< **ActiveMQDestination** > &**originalDestination**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual const **Pointer**< **TransactionId** > & **getOriginalTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getOriginalTransactionId** ()
- virtual void **setOriginalTransactionId** (const **Pointer**< **TransactionId** > &**originalTransactionId**)
- virtual const std::string & **getGroupID** () const
- virtual std::string & **getGroupID** ()
- virtual void **setGroupID** (const std::string &**groupID**)
- virtual int **getGroupSequence** () const
- virtual void **setGroupSequence** (int **groupSequence**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual bool **isPersistent** () const

- virtual void **setPersistent** (bool **persistent**)
- virtual long long **getExpiration** () const
- virtual void **setExpiration** (long long **expiration**)
- virtual unsigned char **getPriority** () const
- virtual void **setPriority** (unsigned char **priority**)
- virtual const **Pointer< ActiveMQDestination >** & **getReplyTo** () const
- virtual **Pointer< ActiveMQDestination >** & **getReplyTo** ()
- virtual void **setReplyTo** (const **Pointer< ActiveMQDestination >** &**replyTo**)
- virtual long long **getTimestamp** () const
- virtual void **setTimestamp** (long long **timestamp**)
- virtual const std::string & **getType** () const
- virtual std::string & **getType** ()
- virtual void **setType** (const std::string &**type**)
- virtual const std::vector< unsigned char > & **getContent** () const
- virtual std::vector< unsigned char > & **getContent** ()
- virtual void **setContent** (const std::vector< unsigned char > &**content**)
- virtual const std::vector< unsigned char > & **getMarshalledProperties** () const
- virtual std::vector< unsigned char > & **getMarshalledProperties** ()
- virtual void **setMarshalledProperties** (const std::vector< unsigned char > &**marshalledProperties**)
- virtual const **Pointer< DataStructure >** & **getDataStructure** () const
- virtual **Pointer< DataStructure >** & **getDataStructure** ()
- virtual void **setDataStructure** (const **Pointer< DataStructure >** &**dataStructure**)
- virtual const **Pointer< ConsumerId >** & **getTargetConsumerId** () const
- virtual **Pointer< ConsumerId >** & **getTargetConsumerId** ()
- virtual void **setTargetConsumerId** (const **Pointer< ConsumerId >** &**targetConsumerId**)
- virtual bool **isCompressed** () const
- virtual void **setCompressed** (bool **compressed**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId >** > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId >** > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer< BrokerId >** > &**brokerPath**)
- virtual long long **getArrival** () const
- virtual void **setArrival** (long long **arrival**)
- virtual const std::string & **getUserID** () const
- virtual std::string & **getUserID** ()
- virtual void **setUserID** (const std::string &**userID**)
- virtual bool **isRecievedByDFBridge** () const
- virtual void **setRecievedByDFBridge** (bool **recievedByDFBridge**)
- virtual bool **isDroppable** () const
- virtual void **setDroppable** (bool **droppable**)
- virtual const std::vector< **decaf::lang::Pointer< BrokerId >** > & **getCluster** () const
- virtual std::vector< **decaf::lang::Pointer< BrokerId >** > & **getCluster** ()
- virtual void **setCluster** (const std::vector< **decaf::lang::Pointer< BrokerId >** > &**cluster**)
- virtual long long **getBrokerInTime** () const

- virtual void **setBrokerInTime** (long long **brokerInTime**)
- virtual long long **getBrokerOutTime** () const
- virtual void **setBrokerOutTime** (long long **brokerOutTime**)
- virtual bool **isMessage** () const
- virtual **Pointer< Command > visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGE** = 0

Protected Member Functions

- **Message** (const **Message** &)
- **Message** & **operator=** (const **Message** &)

Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ActiveMQDestination > originalDestination**
- **Pointer< MessageId > messageId**
- **Pointer< TransactionId > originalTransactionId**
- std::string **groupID**
- int **groupSequence**
- std::string **correlationId**
- bool **persistent**
- long long **expiration**
- unsigned char **priority**
- **Pointer< ActiveMQDestination > replyTo**
- long long **timestamp**
- std::string **type**
- std::vector< unsigned char > **content**
- std::vector< unsigned char > **marshalledProperties**
- **Pointer< DataStructure > dataStructure**
- **Pointer< ConsumerId > targetConsumerId**
- bool **compressed**
- int **redeliveryCounter**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- long long **arrival**
- std::string **userID**
- bool **recievedByDFBridge**
- bool **droppable**
- std::vector< **decaf::lang::Pointer< BrokerId > > cluster**
- long long **brokerInTime**
- long long **brokerOutTime**

Static Protected Attributes

- static const unsigned int **DEFAULT_MESSAGE_SIZE** = 1024

6.348.1 Constructor & Destructor Documentation

6.348.1.1 activemq::commands::Message::Message (const Message &) [inline, protected]

6.348.1.2 activemq::commands::Message::Message ()

6.348.1.3 virtual activemq::commands::Message::~Message () [virtual]

6.348.2 Member Function Documentation

6.348.2.1 virtual void activemq::commands::Message::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p. 558).

6.348.2.2 virtual void activemq::commands::Message::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshaled to the wire.

Parameters:

wireFormat - the **wireformat** (p. 74) controller

Reimplemented from **activemq::commands::BaseDataStructure** (p. 558).

6.348.2.3 virtual Message* activemq::commands::Message::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 147), **activemq::commands::ActiveMQBytesMessage** (p. 166),

activemq::commands::ActiveMQMapMessage (p. 258), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 377), and **activemq::commands::ActiveMQTextMessage** (p. 450).

6.348.2.4 virtual void activemq::commands::Message::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 147), **activemq::commands::ActiveMQBytesMessage** (p. 167), **activemq::commands::ActiveMQMapMessage** (p. 258), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 377), and **activemq::commands::ActiveMQTextMessage** (p. 450).

6.348.2.5 virtual bool activemq::commands::Message::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 148), **activemq::commands::ActiveMQBytesMessage** (p. 167), **activemq::commands::ActiveMQMapMessage** (p. 259), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 377), and **activemq::commands::ActiveMQTextMessage** (p. 450).

6.348.2.6 virtual core::ActiveMQAckHandler* activemq::commands::Message::getAckHandler () const [inline, virtual]

Gets the Acknowledgment Handler that this **Message** (p. 1737) will use when the Acknowledge method is called.

Returns:

handler ActiveMQAckHandler to call or NULL if not set

6.348.2.7 virtual long long activemq::commands::Message::getArrival () const [virtual]

6.348.2.8 virtual long long activemq::commands::Message::getBrokerInTime () const [virtual]

6.348.2.9 virtual long long activemq::commands::Message::getBrokerOutTime () const [virtual]

6.348.2.10 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () [virtual]

6.348.2.11 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getBrokerPath () const [virtual]

6.348.2.12 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () [virtual]

6.348.2.13 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::Message::getCluster () const [virtual]

6.348.2.14 virtual std::vector<unsigned char>& activemq::commands::Message::getContent () [virtual]

6.348.2.15 virtual const std::vector<unsigned char>& activemq::commands::Message::getContent () const [virtual]

6.348.2.16 virtual std::string& activemq::commands::Message::getCorrelationId () [virtual]

6.348.2.17 virtual const std::string& activemq::commands::Message::getCorrelationId () const [virtual]

6.348.2.18 virtual Pointer<DataStructure>& activemq::commands::Message::getDataStructure () [virtual]

6.348.2.19 virtual const Pointer<DataStructure>& activemq::commands::Message::getDataStructure () const [virtual]

6.348.2.20 virtual unsigned char activemq::commands::Message::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 148), **activemq::commands::ActiveMQBytesMessage** (p. 168), **activemq::commands::ActiveMQMapMessage** (p. 260), **activemq::commands::ActiveMQMessage** (p. 280), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQStreamMessage** (p. 377), and **activemq::commands::ActiveMQTextMessage** (p. 451).

6.348.2.21 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () [virtual]

6.348.2.22 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getDestination () const [virtual]

6.348.2.23 virtual long long activemq::commands::Message::getExpiration () const [virtual]

6.348.2.24 virtual std::string& activemq::commands::Message::getGroupID () [virtual]

6.348.2.25 virtual const std::string& activemq::commands::Message::getGroupID () const [virtual]

6.348.2.26 virtual int activemq::commands::Message::getGroupSequence () const [virtual]

6.348.2.27 virtual std::vector<unsigned char>& activemq::commands::Message::getMarshalledProperties () [virtual]

6.348.2.28 virtual const std::vector<unsigned char>& activemq::commands::Message::getMarshalledProperties () const [virtual]

6.348.2.29 virtual Pointer<MessageId>& activemq::commands::Message::getMessageId () [virtual]

6.348.2.30 virtual const Pointer<MessageId>& activemq::commands::Message::getMessageId () const [virtual]

6.348.2.31 const util::PrimitiveMap& activemq::commands::Message::getMessageProperties () const [inline]

6.348.2.32 util::PrimitiveMap& activemq::commands::Message::getMessageProperties () [inline]

Gets a reference to the Message's Properties object, allows the derived classes to get and set their own specific properties.

Returns:

a reference to the Primitive Map that holds message properties.

6.348.2.33 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () [virtual]

6.348.2.34 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getOriginalDestination () const [virtual]

6.348.2.35 virtual Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () [virtual]

6.348.2.36 virtual const Pointer<TransactionId>& activemq::commands::Message::getOriginalTransactionId () const [virtual]

6.348.2.37 virtual unsigned char activemq::commands::Message::getPriority () const [virtual]

6.348.2.38 virtual Pointer<ProducerId>& activemq::commands::Message::getProducerId () [virtual]

6.348.2.39 virtual const Pointer<ProducerId>& activemq::commands::Message::getProducerId () const [virtual]

6.348.2.40 virtual int activemq::commands::Message::getRedeliveryCounter () const [virtual]

6.348.2.41 virtual Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () [virtual]

6.348.2.42 virtual const Pointer<ActiveMQDestination>& activemq::commands::Message::getReplyTo () const [virtual]

6.348.2.43 virtual unsigned int activemq::commands::Message::getSize () const [virtual]

Returns the Size of this message in Bytes.

Returns:

number of bytes this message equates to.

6.348.2.44 virtual Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () [virtual]

6.348.2.45 virtual const Pointer<ConsumerId>& activemq::commands::Message::getTargetConsumerId () const [virtual]

6.348.2.46 virtual long long activemq::commands::Message::getTimestamp () const [virtual]

6.348.2.47 virtual Pointer<TransactionId>& activemq::commands::Message::getTransactionId () [virtual]

6.348.2.48 virtual const Pointer<TransactionId>& activemq::commands::Message::getTransactionId () const [virtual]

6.348.2.49 virtual std::string& activemq::commands::Message::getType () [virtual]

6.348.2.50 virtual const std::string& activemq::commands::Message::getType () const [virtual]

6.348.2.51 virtual std::string& activemq::commands::Message::getUserID () [virtual]

6.348.2.52 virtual const std::string& activemq::commands::Message::getUserID () const [virtual]

6.348.2.53 virtual bool activemq::commands::Message::isCompressed () const [virtual]

6.348.2.54 virtual bool activemq::commands::Message::isDroppable () const [virtual]

6.348.2.55 virtual bool activemq::commands::Message::isExpired () const [inline, virtual]

Returns if this message has expired, meaning that its Expiration time has elapsed.

Returns:

true if message is expired.

References decaf::util::Date::getCurrentTimeMilliseconds().

6.348.2.56 virtual bool activemq::commands::Message::isMarshalAware () const [inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns:

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 559).

Reimplemented in **activemq::commands::ActiveMQMapMessage** (p. 262), and **activemq::commands::ActiveMQStreamMessage** (p. 378).

6.348.2.57 virtual bool activemq::commands::Message::isMessage () const [inline, virtual]

Returns:

an answer of true to the **isMessage()** (p. 1747) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.348.2.58 virtual bool activemq::commands::Message::isPersistent () const [virtual]

6.348.2.59 bool activemq::commands::Message::isReadOnlyBody () const [inline]

Returns if the **Message** (p. 1737) Body is Read Only.

Returns:

true if **Message** (p. 1737) Content is Read Only.

6.348.2.60 bool activemq::commands::Message::isReadOnlyProperties () const [inline]

Returns if the **Message** (p. 1737) Properties Are Read Only.

Returns:

true if **Message** (p. 1737) Properties are Read Only.

6.348.2.61 virtual bool activemq::commands::Message::isRecievedByDFBridge () const [virtual]

6.348.2.62 Message& activemq::commands::Message::operator= (const Message &) [inline, protected]

6.348.2.63 virtual void activemq::commands::Message::setAckHandler (core::ActiveMQAckHandler * *handler*) [inline, virtual]

Sets the Acknowledgment Handler that this **Message** (p. 1737) will use when the Acknowledge method is called.

Parameters:

handler ActiveMQAckHandler to call

6.348.2.64 virtual void activemq::commands::Message::setArrival (long long *arrival*) [virtual]

6.348.2.65 virtual void activemq::commands::Message::setBrokerInTime (long long *brokerInTime*) [virtual]

6.348.2.66 virtual void activemq::commands::Message::setBrokerOutTime (long long *brokerOutTime*) [virtual]

6.348.2.67 virtual void activemq::commands::Message::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.348.2.68 virtual void activemq::commands::Message::setCluster (const std::vector< decaf::lang::Pointer< BrokerId > > & *cluster*) [virtual]

6.348.2.69 virtual void activemq::commands::Message::setCompressed (bool *compressed*) [virtual]

6.348.2.70 virtual void activemq::commands::Message::setContent (const std::vector< unsigned char > & *content*) [virtual]

6.348.2.71 virtual void activemq::commands::Message::setCorrelationId (const std::string & *correlationId*) [virtual]

6.348.2.72 virtual void activemq::commands::Message::setDataStructure (const Pointer< DataStructure > & *dataStructure*) [virtual]

6.348.2.73 virtual void activemq::commands::Message::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.348.2.74 virtual void activemq::commands::Message::setDroppable (bool *droppable*) [virtual]

6.348.2.75 virtual void activemq::commands::Message::setExpiration (long long *expiration*) [virtual]

6.348.2.76 virtual void activemq::commands::Message::setGroupID (const std::string & *groupID*) [virtual]

6.348.2.77 virtual void activemq::commands::Message::setGroupSequence (int *groupSequence*) [virtual]

6.348.2.78 virtual void activemq::commands::Message::setMarshalledProperties (const std::vector< unsigned char > & *marshalledProperties*) [virtual]

6.348.2.79 virtual void activemq::commands::Message::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.348.2.80 virtual void activemq::commands::Message::setOriginalDestination (const Pointer< ActiveMQDestination > & *originalDestination*) [virtual]

6.348.2.81 virtual void activemq::commands::Message::setOriginalTransactionId (const Pointer< TransactionId > & *originalTransactionId*) [virtual]

6.348.2.82 virtual void activemq::commands::Message::setPersistent (bool *persistent*) [virtual]

6.348.2.83 virtual void activemq::commands::Message::setPriority (unsigned char *priority*) [virtual]

Parameters:

value - true if Content should be read only.

6.348.2.86 void activemq::commands::Message::setReadOnlyProperties (bool *value*) [inline]

Set the Read Only State of the **Message** (p. 1737) Properties.

Parameters:

value - true if Properties should be read only.

6.348.2.87 virtual void activemq::commands::Message::setRecievedByDFBridge (bool *recievedByDFBridge*) [virtual]

6.348.2.88 virtual void activemq::commands::Message::setRedeliveryCounter (int *redeliveryCounter*) [virtual]

6.348.2.89 virtual void activemq::commands::Message::setReplyTo (const Pointer< ActiveMQDestination > & *replyTo*) [virtual]

6.348.2.90 virtual void activemq::commands::Message::setTargetConsumerId (const Pointer< ConsumerId > & *targetConsumerId*) [virtual]

6.348.2.91 virtual void activemq::commands::Message::setTimestamp (long long *timestamp*) [virtual]

6.348.2.92 virtual void activemq::commands::Message::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.348.2.93 virtual void activemq::commands::Message::setType (const std::string & *type*) [virtual]

6.348.2.94 virtual void activemq::commands::Message::setUserID (const std::string & *userID*) [virtual]

6.348.2.95 virtual std::string activemq::commands::Message::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

Reimplemented in **activemq::commands::ActiveMQBlobMessage** (p. 150), **activemq::commands::ActiveMQBytesMessage** (p. 173), **activemq::commands::ActiveMQMapMessage** (p. 266), **activemq::commands::ActiveMQMessage** (p. 281), **activemq::commands::ActiveMQObjectMessage** (p. 312), **activemq::commands::ActiveMQStreamMessage** (p. 382), and **activemq::commands::ActiveMQTextMessage** (p. 452).

6.348.2.96 virtual Pointer<Command> activemq::commands::Message::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.348.3 Field Documentation

6.348.3.1 long long activemq::commands::Message::arrival [protected]

6.348.3.2 long long activemq::commands::Message::brokerInTime [protected]

6.348.3.3 long long activemq::commands::Message::brokerOutTime [protected]

6.348.3.4 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::brokerPath [protected]

6.348.3.5 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::Message::cluster [protected]

6.348.3.6 bool activemq::commands::Message::compressed [protected]

6.348.3.7 std::vector<unsigned char> activemq::commands::Message::content [protected]

6.348.3.8 std::string activemq::commands::Message::correlationId [protected]

6.348.3.9 Pointer<DataStructure> activemq::commands::Message::dataStructure [protected]

6.348.3.10 const unsigned int activemq::commands::Message::DEFAULT_MESSAGE_SIZE = 1024 [static, protected]

6.348.3.11 Pointer<ActiveMQDestination> activemq::commands::Message::destination [protected]

6.348.3.12 bool activemq::commands::Message::droppable [protected]

6.348.3.13 long long activemq::commands::Message::expiration [protected]

6.348.3.14 std::string activemq::commands::Message::groupID [protected]

6.348.3.15 int activemq::commands::Message::groupSequence [protected]

6.348.3.16 const unsigned char activemq::commands::Message::ID_MESSAGE = 0 [static]

6.348.3.17 std::vector<unsigned char> activemq::commands::Message::marshalledProperties [protected]

6.348.3.18 Pointer<MessageId> activemq::commands::Message::messageId [protected]

6.348.3.19 Pointer<ActiveMQDestination> activemq::commands::Message::originalDestination [protected]

6.348.3.20 Pointer<TransactionId> activemq::commands::Message::originalTransactionId [protected]

6.348.3.21 bool activemq::commands::Message::persistent [protected]

6.348.3.22 unsigned char activemq::commands::Message::priority [protected]

6.348.3.23 Pointer<ProducerId> activemq::commands::Message::producerId [protected]

- src/main/activemq/commands/**Message.h**

6.349 cms::Message Class Reference

Root of all messages.

`#include <src/main/cms/Message.h>`Inheritance diagram for cms::Message:

Public Member Functions

- virtual ~**Message** ()
- virtual **Message** * **clone** () const =0

  Clone this message exactly, returns a new instance that the caller is required to delete.

- virtual void **acknowledge** () const =0 throw (CMSException)

  Acknowledges all consumed messages of the session of this consumed message.

- virtual void **clearBody** ()=0 throw (CMSException)

  Clears out the body of the message.

- virtual void **clearProperties** ()=0 throw (CMSException)

  Clears out the message body.

- virtual std::vector< std::string > **getPropertyNames** () const =0 throw (CMSException)

  Retrieves the property names.

- virtual bool **propertyExists** (const std::string &name) const =0 throw (CMSException)

  Indicates whether or not a given property exists.

- virtual bool **getBooleanProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a byte property.

- virtual double **getDoubleProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a double property.

- virtual float **getFloatProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const =0 throw (CMSException)

  Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value)=0 throw (CMSException)

  Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value)=0 throw (CMSException)

  Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value)=0 throw (CMSException)

  Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value)=0 throw (CMSException)

  Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value)=0 throw (CMSException)

  Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value)=0 throw (CMSException)

  Sets a long property.

- virtual void **setShortProperty** (const std::string &name, short value)=0 throw (CMSException)

  Sets a short property.

- virtual void **setStringProperty** (const std::string &name, const std::string &value)=0 throw (CMSException)

  Sets a string property.

- virtual std::string **getCMSCorrelationID** () const =0 throw (CMSException)

  Gets the correlation ID for the message.

- virtual void **setCMSCorrelationID** (const std::string &correlationId)=0 throw (CMSException)

  Sets the correlation ID for the message.

- virtual int **getCMSDeliveryMode** () const =0 throw (CMSException)

  *Gets the **DeliveryMode** (p. 1188) for this message.*

- virtual void **setCMSDeliveryMode** (int mode)=0 throw (CMSException)

  *Sets the **DeliveryMode** (p. 1188) for this message.*

- virtual const **Destination** * **getCMSDestination** () const =0 throw (CMSException)

  *Gets the **Destination** (p. 1190) object for this message.*

- virtual void **setCMSDestination** (const **Destination** *destination)=0 throw (CMSException)

  *Sets the **Destination** (p. 1190) object for this message.*

- virtual long long **getCMSExpiration** () const =0 throw (CMSException)

  Gets the message's expiration value.

- virtual void **setCMSExpiration** (long long expireTime)=0 throw (CMSException)

  Sets the message's expiration value.

- virtual std::string **getCMSMessageID** () const =0 throw (CMSException)

  The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider.

- virtual void **setCMSMessageID** (const std::string &id)=0 throw (CMSException)

  Sets the message ID.

- virtual int **getCMSPriority** () const =0 throw (CMSException)

  Gets the message priority level.

- virtual void **setCMSPriority** (int priority)=0 throw (CMSException)

  Sets the Priority Value for this message.

- virtual bool **getCMSRedelivered** () const =0 throw (CMSException)

  Gets an indication of whether this message is being redelivered.

- virtual void **setCMSRedelivered** (bool redelivered)=0 throw (CMSException)

  Specifies whether this message is being redelivered.

- virtual const **cms::Destination** * **getCMSReplyTo** () const =0 throw (CMSException)

  *Gets the **Destination** (p. 1190) object to which a reply to this message should be sent.*

- virtual void **setCMSReplyTo** (const **cms::Destination** *destination)=0 throw (CMSException)

  *Sets the **Destination** (p. 1190) object to which a reply to this message should be sent.*

- virtual long long **getCMSTimestamp** () const =0 throw (CMSException)

  Gets the message timestamp.

- virtual void **setCMSTimestamp** (long long timeStamp)=0 throw (CMSException)

Sets the message timestamp.

- virtual std::string **getCMSType** () const =0 throw (CMSException)

  Gets the message type identifier supplied by the client when the message was sent.

- virtual void **setCMSType** (const std::string &type)=0 throw (CMSException)

  Sets the message type.

6.349.1 Detailed Description

Root of all messages. As in JMS, a message is comprised of 3 parts: CMS-specific headers, user-defined properties, and the body.

Message (p. 1753) Bodies

The CMS API defines four types of message bodies, each type is contained within its own **Message** (p. 1753) Interface definition.

- Stream - A **StreamMessage** (p. 2476) object's message body contains a stream of primitive values in the C++ language. It is filled and read sequentially. Unlike the **BytesMessage** (p. 759) type the values written to a **StreamMessage** (p. 2476) retain information on their type and rules for type conversion are enforced when reading back the values from the **Message** (p. 1753) Body.
- Map - A **MapMessage** (p. 1701) object's message body contains a set of name-value pairs, where names are std::string objects, and values are C++ primitives. The entries can be accessed sequentially or randomly by name. The **MapMessage** (p. 1701) makes no garuntee on the order of the elements within the **Message** (p. 1753) body.
- Text - A **TextMessage** (p. 2542) object's message body contains a std::string object. This message type can be used to transport plain-text messages, and XML messages.
- Bytes - A **BytesMessage** (p. 759) object's message body contains a stream of uninterpreted bytes. This message type is for literally encoding a body to match an existing message format. In many cases, it is possible to use one of the other body types, which are easier to use.

Message (p. 1753) Properties

Message (p. 1753) properties support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 850). The String-to-primitive conversions may throw a runtime exception if the primitive's valueOf method does not accept the String as a valid representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	int	long	float	double	String
boolean	X							X
byte		X	X	X	X			X
short			X	X	X			X
int				X	X			X
long					X			X
float						X	X	X
double							X	X
String	X	X	X	X	X	X	X	X

|----------

When a **Message** (p. 1753) is delivered its properties are considered to be in a read-only mode and cannot be changed. Attempting to change the value of a delivered Message's properties will result in a **CMSException** (p. 850) being thrown.

See also:

JMS API

Since:

1.0

6.349.2 Constructor & Destructor Documentation

6.349.2.1 virtual cms::Message::~Message () [inline, virtual]

6.349.3 Member Function Documentation

6.349.3.1 virtual void cms::Message::acknowledge () const throw (CMSException) [pure virtual]

Acknowledges all consumed messages of the session of this consumed message. All consumed CMS messages support the acknowledge method for use when a client has specified that its CMS session's consumed messages are to be explicitly acknowledged. By invoking acknowledge on a consumed message, a client acknowledges all messages consumed by the session that the message was delivered to.

Calls to acknowledge are ignored for both transacted sessions and sessions specified to use implicit acknowledgment modes.

A client may individually acknowledge each message as it is consumed, or it may choose to acknowledge messages as an application-defined group (which is done by calling acknowledge on the last received message of the group, thereby acknowledging all messages consumed by the session.)

Messages that have been received but not acknowledged may be redelivered.

Exceptions:

CMSException *(p. **850**)* - if an internal error occurs.
IllegalStateException *(p. **1399**)* - if this method is called on a closed session.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297).

6.349.3.2 virtual void cms::Message::clearBody () throw (CMSException) [pure virtual]

Clears out the body of the message. This does not clear the headers or properties.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQBytesMessage** (p. 166), **activemq::commands::ActiveMQStreamMessage** (p. 376), **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 297), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 297), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 297).

6.349.3.3 virtual void cms::Message::clearProperties () throw (CMSException) [pure virtual]

Clears out the message body. Clearing a message's body does not clear its header values or property entries.

If this message body was read-only, calling this method leaves the message body in the same state as an empty body in a newly created message.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

6.349.3.4 virtual Message* cms::Message::clone () const [pure virtual]

Clone this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

Implemented in **activemq::commands::ActiveMQBlobMessage** (p. 147), **activemq::commands::ActiveMQBytesMessage** (p. 166), **activemq::commands::ActiveMQMapMessage** (p. 258), **activemq::commands::ActiveMQMessage** (p. 279), **activemq::commands::ActiveMQObjectMessage** (p. 311), **activemq::commands::ActiveMQStreamMessage** (p. 376), **activemq::commands::ActiveMQTextMessage** (p. 450), and **cms::BytesMessage** (p. 762).

6.349.3.5 virtual bool cms::Message::getBooleanProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException *(*p. ***850****)* if the property does not exist.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

6.349.3.6 virtual unsigned char cms::Message::getByteProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException *(*p. ***850****)* if the property does not exist.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 298), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 298), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 298).

6.349.3.7 virtual std::string cms::Message::getCMSCorrelationID () const throw (CMSException) [pure virtual]

Gets the correlation ID for the message. This method is used to return correlation ID values that are either provider-specific message IDs or application-specific String values.

Returns:

string representation of the correlation Id

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 298), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 298), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 298), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 298).

6.349.3.8 virtual int cms::Message::getCMSDeliveryMode () const throw (CMSException) [pure virtual]

Gets the **DeliveryMode** (p. 1188) for this message.

Returns:

DeliveryMode (p. 1188) enumerated value.

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 299), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 299).

6.349.3.9 virtual const Destination* cms::Message::getCMSDestination () const throw (CMSException) [pure virtual]

Gets the **Destination** (p. 1190) object for this message. The CMSDestination header field contains the destination to which the message is being sent.

When a message is sent, this field is ignored. After completion of the send or publish method, the field holds the destination specified by the method.

When a message is received, its CMSDestination value must be equivalent to the value assigned when it was sent.

Returns:

Destination (p. 1190) object

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<**

cms::MapMessage > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 299), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 299).

6.349.3.10 virtual long long cms::Message::getCMSExpiration () const throw (CMSException) [pure virtual]

Gets the message's expiration value. When a message is sent, the CMSExpiration header field is left unassigned. After completion of the send or publish method, it holds the expiration time of the message. This is the sum of the time-to-live value specified by the client and the GMT at the time of the send or publish.

If the time-to-live is specified as zero, CMSExpiration is set to zero to indicate that the message does not expire.

When a message's expiration time is reached, a provider should discard it. The CMS API does not define any form of notification of message expiration.

Clients should not receive messages that have expired; however, the CMS API does not guarantee that this will not happen.

Returns:

the time the message expires, which is the sum of the time-to-live value specified by the client and the GMT at the time of the send

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 299), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 299), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 299).

6.349.3.11 virtual std::string cms::Message::getCMSMessageID () const throw (CMSException) [pure virtual]

The CMSMessageID header field contains a value that uniquely identifies each message sent by a provider. When a message is sent, CMSMessageID can be ignored. When the send or publish method returns, it contains a provider-assigned value.

A CMSMessageID is a String value that should function as a unique key for identifying messages in a historical repository. The exact scope of uniqueness is provider-defined. It should at least cover all messages for a specific installation of a provider, where an installation is some connected set of message routers.

All CMSMessageID values must start with the prefix 'ID:'. Uniqueness of message ID values across different providers is not required.

Since message IDs take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the message ID is not used

by an application. By calling the **MessageProducer.setDisableMessageID** (p. 1883) method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the message ID set to null; if the provider ignores the hint, the message ID must be set to its normal unique value.

Returns:

provider-assigned message id

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 300).

6.349.3.12 virtual int cms::Message::getCMSPriority () const throw (CMSException) [pure virtual]

Gets the message priority level. The CMS API defines ten levels of priority value, with 0 as the lowest priority and 9 as the highest. In addition, clients should consider priorities 0-4 as gradations of normal priority and priorities 5-9 as gradations of expedited priority.

The CMS API does not require that a provider strictly implement priority ordering of messages; however, it should do its best to deliver expedited messages ahead of normal messages.

Returns:

priority value

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 300).

6.349.3.13 virtual bool cms::Message::getCMSRedelivered () const throw (CMSException) [pure virtual]

Gets an indication of whether this message is being redelivered. If a client receives a message with the CMSRedelivered field set, it is likely, but not guaranteed, that this message was delivered earlier but that its receipt was not acknowledged at that time.

Returns:

true if this message is being redelivered

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 300).

6.349.3.14 virtual const cms::Destination* cms::Message::getCMSReplyTo () const throw (CMSException) [pure virtual]

Gets the **Destination** (p. 1190) object to which a reply to this message should be sent.

Returns:

Destination (p. 1190) to which to send a response to this message

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 300), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 300), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 300).

6.349.3.15 virtual long long cms::Message::getCMSTimestamp () const throw (CMSException) [pure virtual]

Gets the message timestamp. The CMSTimestamp header field contains the time a message was handed off to a provider to be sent. It is not the time the message was actually transmitted, because the actual send may occur later due to transactions or other client-side queuing of messages.

When a message is sent, CMSTimestamp is ignored. When the send or publish method returns, it contains a time value somewhere in the interval between the call and the return. The value is in the format of a normal millis time value in the Java programming language.

Since timestamps take some effort to create and increase a message's size, some CMS providers may be able to optimize message overhead if they are given a hint that the timestamp is not used by an application. By calling the MessageProducer.setDisableMessageTimestamp method, a CMS client enables this potential optimization for all messages sent by that message producer. If the CMS provider accepts this hint, these messages must have the timestamp set to zero; if the provider ignores the hint, the timestamp must be set to its normal value.

Returns:

the message timestamp

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 301), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 301).

6.349.3.16 virtual std::string cms::Message::getCMSType () const throw (CMSException) [pure virtual]

Gets the message type identifier supplied by the client when the message was sent.

Returns:

the message type

See also:

setCMSType (p. 1773)

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 301), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 301).

6.349.3.17 virtual double cms::Message::getDoubleProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException *(*p. ***850)*** if the property does not exist.

MessageFormatException *(*p. ***1842)*** - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 301), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 301), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 301).

6.349.3.18 virtual float cms::Message::getFloatProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException *(*p. ***850)*** if the property does not exist.

MessageFormatException *(*p. ***1842)*** - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 302), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 302), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 302), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 302), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 302), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 302).

6.349.3.19 virtual int cms::Message::getIntProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException *(*p. ***850)*** if the property does not exist.

MessageFormatException *(*p. ***1842)*** - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::Message >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >** (p. 302), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >** (p. 302).

6.349.3.20 virtual long long cms::Message::getLongProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException (p. ***850***) if the property does not exist.

MessageFormatException (p. ***1842***) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::Message >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 302), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >** (p. 302), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >** (p. 302).

6.349.3.21 virtual std::vector<std::string> cms::Message::getPropertyNames () const throw (CMSException) [pure virtual]

Retrieves the property names.

Returns:

The complete set of property names currently in this message.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::Message >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >** (p. 303), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >** (p. 303).

6.349.3.22 virtual short cms::Message::getShortProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException (p. ***850***) if the property does not exist.

MessageFormatException (p. ***1842***) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 303), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 303).

6.349.3.23 virtual std::string cms::Message::getStringProperty (const std::string & *name*) const throw (CMSException) [pure virtual]

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException (p. ***850***) if the property does not exist.

MessageFormatException (p. ***1842***) - if this type conversion is invalid.

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 303), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 303).

6.349.3.24 virtual bool cms::Message::propertyExists (const std::string & *name*) const throw (CMSException) [pure virtual]

Indicates whether or not a given property exists.

Parameters:

name The name of the property to look up.

Returns:

True if the property exists in this message.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::Message >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 303), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >** (p. 303), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >** (p. 303).

6.349.3.25 virtual void cms::Message::setBooleanProperty (const std::string & *name*, bool *value*) throw (CMSException) [pure virtual]

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. ***850***) - if the name is an empty string.

MessageNotWriteableException (p. ***1877***) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate<cms::BytesMessage >** (p. 304), **activemq::commands::ActiveMQMessageTemplate<cms::MapMessage >** (p. 304), **activemq::commands::ActiveMQMessageTemplate<cms::Message >** (p. 304), **activemq::commands::ActiveMQMessageTemplate<cms::StreamMessage >** (p. 304), **activemq::commands::ActiveMQMessageTemplate<cms::TextMessage >** (p. 304), and **activemq::commands::ActiveMQMessageTemplate<cms::ObjectMessage >** (p. 304).

6.349.3.26 virtual void cms::Message::setByteProperty (const std::string & *name*, unsigned char *value*) throw (CMSException) [pure virtual]

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException *(*p. ***850****)* - if the name is an empty string.

MessageNotWriteableException *(*p. ***1877****)* - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 304), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 304), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 304), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 304), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 304), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 304).

6.349.3.27 virtual void cms::Message::setCMSCorrelationID (const std::string & *correlationId*) throw (CMSException) [pure virtual]

Sets the correlation ID for the message. A client can use the CMSCorrelationID header field to link one message with another. A typical use is to link a response message with its request message.

CMSCorrelationID can hold one of the following:

- A provider-specific message ID
- An application-specific String
- A provider-native byte[] value

Since each message sent by a CMS provider is assigned a message ID value, it is convenient to link messages via message ID. All message ID values must start with the 'ID:' prefix.

In some cases, an application (made up of several clients) needs to use an application-specific value for linking messages. For instance, an application may use CMSCorrelationID to hold a value referencing some external information. Application-specified values must not start with the 'ID:' prefix; this is reserved for provider-generated message ID values.

If a provider supports the native concept of correlation ID, a CMS client may need to assign specific CMSCorrelationID values to match those expected by clients that do not use the CMS API. A byte[] value is used for this purpose. CMS providers without native correlation ID values are not required to support byte[] values. The use of a byte[] value for CMSCorrelationID is non-portable.

Parameters:

correlationId The message ID of a message being referred to.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 304), **activemq::commands::ActiveMQMessageTemplate<**

cms::MapMessage > (p. 304), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 304), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 304), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 304), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 304).

6.349.3.28 virtual void cms::Message::setCMSDeliveryMode (int *mode*) throw (CMSException) [pure virtual]

Sets the **DeliveryMode** (p. 1188) for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

mode **DeliveryMode** (p. 1188) enumerated value.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 305), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 305).

6.349.3.29 virtual void cms::Message::setCMSDestination (const Destination * *destination*) throw (CMSException) [pure virtual]

Sets the **Destination** (p. 1190) object for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

destination **Destination** (p. 1190) Object

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 305), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 305).

6.349.3.30 virtual void cms::Message::setCMSExpiration (long long *expireTime*) throw (CMSException) [pure virtual]

Sets the message's expiration value. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

expireTime the message's expiration time

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 305), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 305), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 305), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 305).

6.349.3.31 virtual void cms::Message::setCMSMessageID (const std::string & *id*) throw (CMSException) [pure virtual]

Sets the message ID. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

id the ID of the message

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

6.349.3.32 virtual void cms::Message::setCMSPriority (int *priority*) throw (CMSException) [pure virtual]

Sets the Priority Value for this message. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

priority priority value for this message

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate<**

cms::Message > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 306), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 306).

6.349.3.33 virtual void cms::Message::setCMSRedelivered (bool *redelivered*) throw (CMSException) [pure virtual]

Specifies whether this message is being redelivered. This field is set at the time the message is delivered. This method can be used to change the value for a message that has been received.

Parameters:

redelivered boolean redelivered value

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 306), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 306).

6.349.3.34 virtual void cms::Message::setCMSReplyTo (const cms::Destination * *destination*) throw (CMSException) [pure virtual]

Sets the **Destination** (p. 1190) object to which a reply to this message should be sent. The CMSReplyTo header field contains the destination where a reply to the current message should be sent. If it is null, no reply is expected. The destination may be either a **Queue** (p. 2157) object or a **Topic** (p. 2571) object.

Messages sent with a null CMSReplyTo value may be a notification of some event, or they may just be some data the sender thinks is of interest.

Messages with a CMSReplyTo value typically expect a response. A response is optional; it is up to the client to decide. These messages are called requests. A message sent in response to a request is called a reply.

In some cases a client may wish to match a request it sent earlier with a reply it has just received. The client can use the CMSCorrelationID header field for this purpose.

Parameters:

destination **Destination** (p. 1190) to which to send a response to this message

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate**<

cms::MapMessage > (p. 306), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 306), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 306), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 306), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 306).

6.349.3.35 virtual void cms::Message::setCMSTimestamp (long long *timeStamp*) throw (CMSException) [pure virtual]

Sets the message timestamp. CMS providers set this field when a message is sent. This method can be used to change the value for a message that has been received.

Parameters:

timeStamp integer time stamp value

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 307), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 307).

6.349.3.36 virtual void cms::Message::setCMSType (const std::string & *type*) throw (CMSException) [pure virtual]

Sets the message type. Some CMS providers use a message repository that contains the definitions of messages sent by applications. The CMSType header field may reference a message's definition in the provider's repository.

The CMS API does not define a standard message definition repository, nor does it define a naming policy for the definitions it contains.

Some messaging systems require that a message type definition for each application message be created and that each message specify its type. In order to work with such CMS providers, CMS clients should assign a value to CMSType, whether the application makes use of it or not. This ensures that the field is properly set for those providers that require it.

To ensure portability, CMS clients should use symbolic values for CMSType that can be configured at installation time to the values defined in the current provider's message repository. If string literals are used, they may not be valid type names for some CMS providers.

Parameters:

type the message type

See also:

getCMSType (p. 1764)

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 307), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 307).

6.349.3.37 virtual void cms::Message::setDoubleProperty (const std::string & *name*, double *value*) throw (CMSException) [pure virtual]

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException *(*p. ***850)*** - if the name is an empty string.

MessageNotWriteableException *(*p. ***1877)*** - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::Message** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::StreamMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::TextMessage** > (p. 307), and **activemq::commands::ActiveMQMessageTemplate<** **cms::ObjectMessage** > (p. 307).

6.349.3.38 virtual void cms::Message::setFloatProperty (const std::string & *name*, float *value*) throw (CMSException) [pure virtual]

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException *(*p. ***850)*** - if the name is an empty string.

MessageNotWriteableException *(*p. ***1877)*** - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate<** **cms::BytesMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<** **cms::MapMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate<**

cms::Message > (p. 307), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 307), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 307), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 307).

6.349.3.39 virtual void cms::Message::setIntProperty (const std::string & *name*, int *value*) throw (CMSException) [pure virtual]

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. ***850***) - if the name is an empty string.

MessageNotWriteableException (p. ***1877***) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 308), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 308).

6.349.3.40 virtual void cms::Message::setLongProperty (const std::string & *name*, long long *value*) throw (CMSException) [pure virtual]

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. ***850***) - if the name is an empty string.

MessageNotWriteableException (p. ***1877***) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate**< **cms::BytesMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::MapMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::Message** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::StreamMessage** > (p. 308), **activemq::commands::ActiveMQMessageTemplate**< **cms::TextMessage** > (p. 308), and **activemq::commands::ActiveMQMessageTemplate**< **cms::ObjectMessage** > (p. 308).

6.349.3.41 virtual void cms::Message::setShortProperty (const std::string & *name*, short *value*) throw (CMSException) [pure virtual]

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. ***850***) - if the name is an empty string.

MessageNotWriteableException (p. ***1877***) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 308), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 308), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 308), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 308), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 308), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 308).

6.349.3.42 virtual void cms::Message::setStringProperty (const std::string & *name*, const std::string & *value*) throw (CMSException) [pure virtual]

Sets a string property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException (p. ***850***) - if the name is an empty string.

MessageNotWriteableException (p. ***1877***) - if properties are read-only

Implemented in **activemq::commands::ActiveMQMessageTemplate< cms::BytesMessage >** (p. 309), **activemq::commands::ActiveMQMessageTemplate< cms::MapMessage >** (p. 309), **activemq::commands::ActiveMQMessageTemplate< cms::Message >** (p. 309), **activemq::commands::ActiveMQMessageTemplate< cms::StreamMessage >** (p. 309), **activemq::commands::ActiveMQMessageTemplate< cms::TextMessage >** (p. 309), and **activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >** (p. 309).

The documentation for this class was generated from the following file:

- src/main/cms/**Message.h**

6.350 activemq::commands::MessageAck Class Reference

`#include <src/main/activemq/commands/MessageAck.h>`Inheritance diagram for activemq::commands::MessageAck:

Public Member Functions

- **MessageAck** ()
- virtual ~**MessageAck** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **MessageAck** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual unsigned char **getAckType** () const
- virtual void **setAckType** (unsigned char **ackType**)
- virtual const **Pointer**< **MessageId** > & **getFirstMessageId** () const
- virtual **Pointer**< **MessageId** > & **getFirstMessageId** ()
- virtual void **setFirstMessageId** (const **Pointer**< **MessageId** > &**firstMessageId**)
- virtual const **Pointer**< **MessageId** > & **getLastMessageId** () const
- virtual **Pointer**< **MessageId** > & **getLastMessageId** ()
- virtual void **setLastMessageId** (const **Pointer**< **MessageId** > &**lastMessageId**)
- virtual int **getMessageCount** () const
- virtual void **setMessageCount** (int **messageCount**)
- virtual bool **isMessageAck** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEACK** = 22

Protected Member Functions

- **MessageAck** (const **MessageAck** &)
- **MessageAck** & **operator=** (const **MessageAck** &)

Protected Attributes

- **Pointer< ActiveMQDestination > destination**
- **Pointer< TransactionId > transactionId**
- **Pointer< ConsumerId > consumerId**
- unsigned char **ackType**
- **Pointer< MessageId > firstMessageId**
- **Pointer< MessageId > lastMessageId**
- int **messageCount**

6.350.1 Constructor & Destructor Documentation

6.350.1.1 activemq::commands::MessageAck::MessageAck (const MessageAck &) [inline, protected]

6.350.1.2 activemq::commands::MessageAck::MessageAck ()

6.350.1.3 virtual activemq::commands::MessageAck::~MessageAck () [virtual]

6.350.2 Member Function Documentation

6.350.2.1 virtual MessageAck* activemq::commands::MessageAck::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.350.2.2 virtual void activemq::commands::MessageAck::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.350.2.3 virtual bool activemq::commands::MessageAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.350.2.4 virtual unsigned char activemq::commands::MessageAck::getAckType () const [virtual]

6.350.2.5 virtual Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () [virtual]

6.350.2.6 virtual const Pointer<ConsumerId>& activemq::commands::MessageAck::getConsumerId () const [virtual]

6.350.2.7 virtual unsigned char activemq::commands::MessageAck::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.350.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () [virtual]

6.350.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageAck::getDestination () const [virtual]

6.350.2.10 virtual Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () [virtual]

6.350.2.11 virtual const Pointer<MessageId>& activemq::commands::MessageAck::getFirstMessageId () const [virtual]

6.350.2.12 virtual Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () [virtual]

6.350.2.13 virtual const Pointer<MessageId>& activemq::commands::MessageAck::getLastMessageId () const [virtual]

6.350.2.14 virtual int activemq::commands::MessageAck::getMessageCount () const [virtual]

6.350.2.15 virtual Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () [virtual]

6.350.2.16 virtual const Pointer<TransactionId>& activemq::commands::MessageAck::getTransactionId () const [virtual]

6.350.2.17 virtual bool activemq::commands::MessageAck::isMessageAck () const [inline, virtual]

Returns:

an answer of true to the **isMessageAck()** (p. 1780) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.350.2.18 MessageAck& activemq::commands::MessageAck::operator= (const MessageAck &) [inline, protected]

6.350.2.19 virtual void activemq::commands::MessageAck::setAckType (unsigned char *ackType*) [virtual]

6.350.2.20 virtual void activemq::commands::MessageAck::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.350.2.21 virtual void activemq::commands::MessageAck::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.350.2.22 virtual void activemq::commands::MessageAck::setFirstMessageId (const Pointer< MessageId > & *firstMessageId*) [virtual]

6.350.2.23 virtual void activemq::commands::MessageAck::setLastMessageId (const Pointer< MessageId > & *lastMessageId*) [virtual]

6.350.2.24 virtual void activemq::commands::MessageAck::setMessageCount (int *messageCount*) [virtual]

6.350.2.25 virtual void activemq::commands::MessageAck::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.350.2.26 virtual std::string activemq::commands::MessageAck::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.350.2.27 virtual Pointer<Command> activemq::commands::MessageAck::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.350.3 Field Documentation

6.350.3.1 unsigned char activemq::commands::MessageAck::ackType [protected]

6.350.3.2 Pointer<ConsumerId> activemq::commands::MessageAck::consumerId [protected]

6.350.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageAck::destination [protected]

6.350.3.4 Pointer<MessageId> activemq::commands::MessageAck::firstMessageId [protected]

6.350.3.5 const unsigned char activemq::commands::MessageAck::ID_MESSAGEACK = 22 [static]

6.350.3.6 Pointer<MessageId> activemq::commands::MessageAck::lastMessageId [protected]

6.350.3.7 int activemq::commands::MessageAck::messageCount [protected]

6.350.3.8 Pointer<TransactionId> activemq::commands::MessageAck::transactionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageAck.h**

6.351 activemq::wireformat::openwire::marshal::v2::MessageAckMarshal Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1783).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual ~**MessageAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.351.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1783). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.351.2 Constructor & Destructor Documentation

6.351.2.1 activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::MessageAckMarshal () [inline]

6.351.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::~MessageAckMarsh () [inline, virtual]

6.351.3 Member Function Documentation

6.351.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.351.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.351.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.351.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.351.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.351.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.351.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageAckMarshaller.h**

6.352 activemq::wireformat::openwire::marshal::v3::MessageAckMarshal Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1787).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual ~**MessageAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.352.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1787). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.352.2 Constructor & Destructor Documentation

6.352.2.1 activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::MessageAckMarshal () [inline]

6.352.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::~MessageAckMarsh () [inline, virtual]

6.352.3 Member Function Documentation

6.352.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.352.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.352.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.352.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.352.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.352.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.352.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageAckMarshaller.h**

6.353 activemq::wireformat::openwire::marshal::v1::MessageAckMarshal Class Reference

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1791).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller:

Public Member Functions

- **MessageAckMarshaller** ()
- virtual ~**MessageAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.353.1 Detailed Description

Marshaling code for Open Wire Format for **MessageAckMarshaller** (p. 1791). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.353.2 Constructor & Destructor Documentation

6.353.2.1 activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::MessageAckMarshal () [inline]

6.353.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::~MessageAckMarsh () [inline, virtual]

6.353.3 Member Function Documentation

6.353.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.353.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.353.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.353.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.353.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.353.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.353.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageAckMarshaller.h**

6.354 cms::MessageConsumer Class Reference

A client uses a `MessageConsumer` (p. 1795) to received messages from a destination.

`#include <src/main/cms/MessageConsumer.h>`Inheritance diagram for cms::MessageConsumer:

Public Member Functions

- virtual ~**MessageConsumer** ()
- virtual **Message** * **receive** ()=0 throw (CMSException)

  *Synchronously Receive a **Message** (p. 1753).*

- virtual **Message** * **receive** (int millisecs)=0 throw (CMSException)

  *Synchronously Receive a **Message** (p. 1753), time out after defined interval.*

- virtual **Message** * **receiveNoWait** ()=0 throw (CMSException)

  *Receive a **Message** (p. 1753), does not wait if there isn't a new message to read, returns NULL if nothing read.*

- virtual void **setMessageListener** (**MessageListener** *listener)=0 throw (CMSException)

  *Sets the **MessageListener** (p. 1860) that this class will send notifs on.*

- virtual **MessageListener** * **getMessageListener** () const =0 throw (CMSException)

  *Gets the **MessageListener** (p. 1860) that this class will send mew **Message** (p. 1753) notification events to.*

- virtual std::string **getMessageSelector** () const =0 throw (cms::CMSException)

  Gets this message consumer's message selector expression.

6.354.1 Detailed Description

A client uses a `MessageConsumer` (p. 1795) to received messages from a destination. A client may either synchronously receive a message consumer's messages or have the consumer asynchronously deliver them as they arrive.

For synchronous receipt, a client can request the next message from a message consumer using one of its `receive` methods. There are several variations of `receive` that allow a client to poll or wait for the next message.

For asynchronous delivery, a client can register a `MessageListener` (p. 1860) object with a message consumer. As messages arrive at the message consumer, it delivers them by calling the `MessageListener` (p. 1860)'s `onMessage` method.

When the MessageConsumer's close method is called the method can block while an asynchronous message delivery is in progress or until a `receive` operation completes. A blocked consumer in a `receive` call will return a Null when the close method is called.

See also:

MessageListener (p. 1860)

Since:

1.0

6.354.2 Constructor & Destructor Documentation

6.354.2.1 virtual cms::MessageConsumer::~MessageConsumer () [inline, virtual]

6.354.3 Member Function Documentation

6.354.3.1 virtual MessageListener* cms::MessageConsumer::getMessageListener () const throw (CMSException) [pure virtual]

Gets the **MessageListener** (p. 1860) that this class will send mew **Message** (p. 1753) notification events to.

Returns:

The listener of messages received by this consumer

Exceptions:

CMSException (p. ***850***) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 774), and **activemq::core::ActiveMQConsumer** (p. 224).

6.354.3.2 virtual std::string cms::MessageConsumer::getMessageSelector () const throw (cms::CMSException) [pure virtual]

Gets this message consumer's message selector expression.

Returns:

This Consumer's selector expression or "".

Exceptions:

CMSException (p. ***850***) - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 774), and **activemq::core::ActiveMQConsumer** (p. 224).

6.354.3.3 virtual Message* cms::MessageConsumer::receive (int *millisecs*) throw (CMSException) [pure virtual]

Synchronously Receive a **Message** (p. 1753), time out after defined interval. Returns null if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 774), and **activemq::core::ActiveMQConsumer** (p. 224).

6.354.3.4 virtual Message* cms::MessageConsumer::receive () throw (CMSException) [pure virtual]

Synchronously Receive a **Message** (p. 1753).

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 775), and **activemq::core::ActiveMQConsumer** (p. 225).

6.354.3.5 virtual Message* cms::MessageConsumer::receiveNoWait () throw (CMSException) [pure virtual]

Receive a **Message** (p. 1753), does not wait if there isn't a new message to read, returns NULL if nothing read.

Returns:

new message which the caller owns and must delete.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 775), and **activemq::core::ActiveMQConsumer** (p. 225).

6.354.3.6 virtual void cms::MessageConsumer::setMessageListener (MessageListener * *listener*) throw (CMSException) [pure virtual]

Sets the **MessageListener** (p. 1860) that this class will send notifs on.

Parameters:

listener The listener of messages received by this consumer.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::CachedConsumer** (p. 775), and **activemq::core::ActiveMQConsumer** (p. 226).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageConsumer.h**

6.355 activemq::cmsutil::MessageCreator Class Reference

Creates the user-defined message to be sent by the `CmsTemplate` (p. 858).

`#include <src/main/activemq/cmsutil/MessageCreator.h>`

Public Member Functions

- virtual ~**MessageCreator** ()
- virtual **cms::Message** * **createMessage** (**cms::Session** *session)=0 throw (cms::CMSException)

  Creates a message from the given session.

6.355.1 Detailed Description

Creates the user-defined message to be sent by the `CmsTemplate` (p. 858).

6.355.2 Constructor & Destructor Documentation

6.355.2.1 virtual activemq::cmsutil::MessageCreator::~MessageCreator () [inline, virtual]

6.355.3 Member Function Documentation

6.355.3.1 virtual cms::Message* activemq::cmsutil::MessageCreator::createMessage (cms::Session * *session*) throw (cms::CMSException) [pure virtual]

Creates a message from the given session.

Parameters:

session the CMS `Session`

Exceptions:

cms::CMSException (p. ***850***) if thrown by CMS API methods

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**MessageCreator.h**

6.356 activemq::commands::MessageDispatch Class Reference

`#include <src/main/activemq/commands/MessageDispatch.h>`Inheritance diagram for activemq::commands::MessageDispatch:

Public Member Functions

- **MessageDispatch** ()
- virtual ~**MessageDispatch** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **MessageDispatch** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const **Pointer**< **Message** > & **getMessage** () const
- virtual **Pointer**< **Message** > & **getMessage** ()
- virtual void **setMessage** (const **Pointer**< **Message** > &**message**)
- virtual int **getRedeliveryCounter** () const
- virtual void **setRedeliveryCounter** (int **redeliveryCounter**)
- virtual bool **isMessageDispatch** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEDISPATCH** = 21

Protected Member Functions

- **MessageDispatch** (const **MessageDispatch** &)
- **MessageDispatch** & **operator**= (const **MessageDispatch** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- **Pointer**< **Message** > **message**
- int **redeliveryCounter**

6.356.1 Constructor & Destructor Documentation

6.356.1.1 activemq::commands::MessageDispatch::MessageDispatch (const MessageDispatch &) [inline, protected]

6.356.1.2 activemq::commands::MessageDispatch::MessageDispatch ()

6.356.1.3 virtual activemq::commands::MessageDispatch::~MessageDispatch () [virtual]

6.356.2 Member Function Documentation

6.356.2.1 virtual MessageDispatch* activemq::commands::MessageDispatch::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.356.2.2 virtual void activemq::commands::MessageDispatch::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.356.2.3 virtual bool activemq::commands::MessageDispatch::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.356.2.4 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () [virtual]

6.356.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatch::getConsumerId () const [virtual]

6.356.2.6 virtual unsigned char activemq::commands::MessageDispatch::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.356.2.7 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () [virtual]

6.356.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatch::getDestination () const [virtual]

6.356.2.9 virtual Pointer<Message>& activemq::commands::MessageDispatch::getMessage () [virtual]

6.356.2.10 virtual const Pointer<Message>& activemq::commands::MessageDispatch::getMessage () const [virtual]

6.356.2.11 virtual int activemq::commands::MessageDispatch::getRedeliveryCounter () const [virtual]

6.356.2.12 virtual bool activemq::commands::MessageDispatch::isMessageDispatch () const [inline, virtual]

Returns:

an answer of true to the **isMessageDispatch()** (p. 1803) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.356.2.13 MessageDispatch& activemq::commands::MessageDispatch::operator= (const MessageDispatch &) [inline, protected]

6.356.2.14 virtual void activemq::commands::MessageDispatch::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.356.2.15 virtual void activemq::commands::MessageDispatch::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.356.2.16 virtual void activemq::commands::MessageDispatch::setMessage (const Pointer< Message > & *message*) [virtual]

6.356.2.17 virtual void activemq::commands::MessageDispatch::setRedeliveryCounter (int *redeliveryCounter*) [virtual]

6.356.2.18 virtual std::string activemq::commands::MessageDispatch::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.356.2.19 virtual Pointer<Command> activemq::commands::MessageDispatch::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.356.3 Field Documentation

6.356.3.1 Pointer<ConsumerId> activemq::commands::MessageDispatch::consumerId [protected]

6.356.3.2 Pointer<ActiveMQDestination> activemq::commands::MessageDispatch::destination [protected]

6.356.3.3 const unsigned char activemq::commands::MessageDispatch::ID_MESSAGEDISPATCH = 21 [static]

6.356.3.4 Pointer<Message> activemq::commands::MessageDispatch::message [protected]

6.356.3.5 int activemq::commands::MessageDispatch::redeliveryCounter [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatch.h**

6.357 activemq::core::MessageDispatchChannel Class Reference

`#include <src/main/activemq/core/MessageDispatchChannel.h>`Inheritance diagram for activemq::core::MessageDispatchChannel:

Public Member Functions

- **MessageDispatchChannel** ()
- virtual ~**MessageDispatchChannel** ()
- void **enqueue** (const **Pointer**< **MessageDispatch** > &message)

  Add a Message to the Channel behind all pending message.

- void **enqueueFirst** (const **Pointer**< **MessageDispatch** > &message)

  Add a message to the front of the Channel.

- bool **isEmpty** () const
- bool **isClosed** () const
- bool **isRunning** () const
- **Pointer**< **MessageDispatch** > **dequeue** (long long timeout) throw (exceptions::ActiveMQException)

  Used to get an enqueued message.

- **Pointer**< **MessageDispatch** > **dequeueNoWait** ()

  Used to get an enqueued message if there is one queued right now.

- **Pointer**< **MessageDispatch** > **peek** () const

  Peek in the Queue and return the first message in the Channel without removing it from the channel.

- void **start** ()

  Starts dispatch of messages from the Channel.

- void **stop** ()

  Stops dispatch of message from the Channel.

- void **close** ()

  Close this channel no messages will be dispatched after this method is called.

- void **clear** ()

  Clear the Channel, all pending messages are removed.

- int **size** () const
- std::vector< **Pointer**< **MessageDispatch** > > **removeAll** ()

  Remove all messages that are currently in the Channel and return them as a list of Messages.

- virtual void **lock** () throw (decaf::lang::Exception)

Locks the object.

- virtual void **unlock** () throw (decaf::lang::Exception)
  Unlocks the object.
- virtual void **wait** () throw (decaf::lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs) throw (decaf::lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (decaf::lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (decaf::lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.357.1 Constructor & Destructor Documentation

6.357.1.1 activemq::core::MessageDispatchChannel::MessageDispatchChannel ()

6.357.1.2 virtual activemq::core::MessageDispatchChannel::~MessageDispatchChannel () [virtual]

6.357.2 Member Function Documentation

6.357.2.1 void activemq::core::MessageDispatchChannel::clear ()

Clear the Channel, all pending messages are removed.

6.357.2.2 void activemq::core::MessageDispatchChannel::close ()

Close this channel no messages will be dispatched after this method is called.

6.357.2.3 Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeue (long long *timeout*) throw (exceptions::ActiveMQException)

Used to get an enqueued message. The amount of time this method blocks is based on the timeout value. - if timeout==-1 then it blocks until a message is received. - if timeout==0 then it it tries to not block at all, it returns a message if it is available - if timeout>0 then it blocks up to timeout amount of time. Expired messages will consumed by this method.

Returns:

null if we timeout or if the consumer is closed.

Exceptions:

ActiveMQException

6.357.2.4 Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::dequeueNoWait ()

Used to get an enqueued message if there is one queued right now. If there is no waiting message than this method returns Null.

Returns:

a message if there is one in the queue.

6.357.2.5 void activemq::core::MessageDispatchChannel::enqueue (const Pointer< MessageDispatch > & *message*)

Add a Message to the Channel behind all pending message.

Parameters:

message - The message to add to the Channel.

6.357.2.6 void activemq::core::MessageDispatchChannel::enqueueFirst (const Pointer< MessageDispatch > & *message*)

Add a message to the front of the Channel.

Parameters:

message - The Message to add to the front of the Channel.

6.357.2.7 bool activemq::core::MessageDispatchChannel::isClosed () const [inline]

Returns:

has the Queue been closed.

6.357.2.8 bool activemq::core::MessageDispatchChannel::isEmpty () const

Returns:

true if there are no messages in the Channel.

6.357.2.9 bool activemq::core::MessageDispatchChannel::isRunning () const [inline]

Returns:

true if the Channel currently running and will dequeue message.

6.357.2.10 virtual void activemq::core::MessageDispatchChannel::lock () throw (decaf::lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.357.2.11 virtual void activemq::core::MessageDispatchChannel::notify () throw (decaf::lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.357.2.12 virtual void activemq::core::MessageDispatchChannel::notifyAll () throw (decaf::lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.357.2.13 Pointer<MessageDispatch> activemq::core::MessageDispatchChannel::peek () const

Peek in the Queue and return the first message in the Channel without removing it from the channel.

Returns:

a message if there is one in the queue.

6.357.2.14 std::vector< Pointer<MessageDispatch> > activemq::core::MessageDispatchChannel::removeAll ()

Remove all messages that are currently in the Channel and return them as a list of Messages.

Returns:

a list of Messages that was previously in the Channel.

6.357.2.15 int activemq::core::MessageDispatchChannel::size () const

Returns:

the number of Messages currently in the Channel.

6.357.2.16 void activemq::core::MessageDispatchChannel::start ()

Starts dispatch of messages from the Channel.

6.357.2.17 void activemq::core::MessageDispatchChannel::stop ()

Stops dispatch of message from the Channel.

6.357.2.18 virtual void activemq::core::MessageDispatchChannel::unlock () throw (decaf::lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.357.2.19 virtual void activemq::core::MessageDispatchChannel::wait (unsigned long *millisecs*) throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.357.2.20 virtual void activemq::core::MessageDispatchChannel::wait () throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/activemq/core/**MessageDispatchChannel.h**

6.358 activemq::wireformat::openwire::marshal::v2::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1811).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual ~**MessageDispatchMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.358.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1811). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.358.2 Constructor & Destructor Documentation

6.358.2.1 activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::MessageDispat() [inline]

6.358.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::~MessageDispa() [inline, virtual]

6.358.3 Member Function Documentation

6.358.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.358.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::getDataStructure() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.358.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.358.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.358.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.358.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.358.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchMarshaller.h**

6.359 activemq::wireformat::openwire::marshal::v3::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1815).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual ~**MessageDispatchMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.359.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1815). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.359.2 Constructor & Destructor Documentation

6.359.2.1 activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::MessageDispat() [inline]

6.359.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::~MessageDispa() [inline, virtual]

6.359.3 Member Function Documentation

6.359.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.359.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::getDataStructure() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.359.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.359.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.359.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.359.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.359.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageDispatchMarshaller.h**

6.360 activemq::wireformat::openwire::marshal::v1::MessageDispatchMa Class Reference

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1819).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller:

Public Member Functions

- **MessageDispatchMarshaller** ()
- virtual ~**MessageDispatchMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.360.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchMarshaller** (p. 1819). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.360.2 Constructor & Destructor Documentation

6.360.2.1 activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::MessageDispat() [inline]

6.360.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::~MessageDispa() [inline, virtual]

6.360.3 Member Function Documentation

6.360.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.360.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::getDataStructure() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.360.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.360.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.360.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.360.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.360.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchMarshaller.h**

6.361 activemq::commands::MessageDispatchNotification Class Reference

`#include <src/main/activemq/commands/MessageDispatchNotification.h>`Inheritance diagram for activemq::commands::MessageDispatchNotification:

Public Member Functions

- **MessageDispatchNotification** ()
- virtual ~**MessageDispatchNotification** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **MessageDispatchNotification** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual long long **getDeliverySequenceId** () const
- virtual void **setDeliverySequenceId** (long long **deliverySequenceId**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual bool **isMessageDispatchNotification** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEDISPATCHNOTIFICATION** = 90

Protected Member Functions

- **MessageDispatchNotification** (const **MessageDispatchNotification** &)
- **MessageDispatchNotification** & **operator=** (const **MessageDispatchNotification** &)

Protected Attributes

- **Pointer< ConsumerId > consumerId**
- **Pointer< ActiveMQDestination > destination**
- long long **deliverySequenceId**
- **Pointer< MessageId > messageId**

6.361.1 Constructor & Destructor Documentation

6.361.1.1 activemq::commands::MessageDispatchNotification::MessageDispatchNotification (const MessageDispatchNotification &) [inline, protected]

6.361.1.2 activemq::commands::MessageDispatchNotification::MessageDispatchNotification ()

6.361.1.3 virtual activemq::commands::MessageDispatchNotification::~MessageDispatchNotification () [virtual]

6.361.2 Member Function Documentation

6.361.2.1 virtual MessageDispatchNotification* activemq::commands::MessageDispatchNotification::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.361.2.2 virtual void activemq::commands::MessageDispatchNotification::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.361.2.3 virtual bool activemq::commands::MessageDispatchNotification::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.361.2.4 virtual Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () [virtual]

6.361.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessageDispatchNotification::getConsumerId () const [virtual]

6.361.2.6 virtual unsigned char activemq::commands::MessageDispatchNotification::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.361.2.7 virtual long long activemq::commands::MessageDispatchNotification::getDeliverySequenceId () const [virtual]

6.361.2.8 virtual Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () [virtual]

6.361.2.9 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessageDispatchNotification::getDestination () const [virtual]

6.361.2.10 virtual Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () [virtual]

6.361.2.11 virtual const Pointer<MessageId>& activemq::commands::MessageDispatchNotification::getMessageId () const [virtual]

6.361.2.12 virtual bool activemq::commands::MessageDispatchNotification::isMessageDispatchNotification () const [inline, virtual]

Returns:

an answer of true to the **isMessageDispatchNotification()** (p. 1826) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 510).

6.361.2.13 MessageDispatchNotification& activemq::commands::MessageDispatchNotification::operator= (const MessageDispatchNotification &) [inline, protected]

6.361.2.14 virtual void activemq::commands::MessageDispatchNotification::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.361.2.15 virtual void activemq::commands::MessageDispatchNotification::setDeliverySequenceId (long long *deliverySequenceId*) [virtual]

6.361.2.16 virtual void activemq::commands::MessageDispatchNotification::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.361.2.17 virtual void activemq::commands::MessageDispatchNotification::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.361.2.18 virtual std::string activemq::commands::MessageDispatchNotification::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.361.2.19 virtual Pointer<Command> activemq::commands::MessageDispatchNotification::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.361.3 Field Documentation

6.361.3.1 Pointer<ConsumerId> activemq::commands::MessageDispatchNotification::consumerId [protected]

6.361.3.2 long long activemq::commands::MessageDispatchNotification::deliverySequenceId [protected]

6.361.3.3 Pointer<ActiveMQDestination> activemq::commands::MessageDispatchNotification::destination [protected]

6.361.3.4 const unsigned char activemq::commands::MessageDispatchNotification::ID_MESSAGEDISPATCHNOTIFICATION = 90 [static]

6.361.3.5 Pointer<MessageId> activemq::commands::MessageDispatchNotification::messageId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageDispatchNotification.h**

6.362 activemq::wireformat::openwire::marshal::v2::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1829).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageDispatchNotificationMarshaller`
diagram for activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.362.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1829). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.362.2 Constructor & Destructor Documentation

6.362.2.1 activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::Me () [inline]

6.362.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::~M () [inline, virtual]

6.362.3 Member Function Documentation

6.362.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::create () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.362.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::getDa () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.362.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.362.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::looseU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.362.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.362.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.362.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller::tightU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageDispatchNotificationMarshaller.h**

6.363 activemq::wireformat::openwire::marshal::v1::MessageDispatchNo Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1833).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchNotificationMarshaller`
diagram for activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.363.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1833). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.363.2 Constructor & Destructor Documentation

6.363.2.1 activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::Me () [inline]

6.363.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::~M () [inline, virtual]

6.363.3 Member Function Documentation

6.363.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::create () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.363.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::getDa () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.363.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.363.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::looseU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.363.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.363.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.363.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller::tightU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageDispatchNotificationMarshaller.h**

6.364 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1837).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchNotificationMarshaller`
diagram for activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller:

Public Member Functions

- **MessageDispatchNotificationMarshaller** ()
- virtual ~**MessageDispatchNotificationMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.364.1 Detailed Description

Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1837). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.364.2 Constructor & Destructor Documentation

6.364.2.1 activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::Me () [inline]

6.364.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::~M () [inline, virtual]

6.364.3 Member Function Documentation

6.364.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::create () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.364.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::getDa () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.364.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.364.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::looseU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.364.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.364.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightM (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.364.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller::tightU (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageDispatchNotificationMarshaller.h**

6.365 cms::MessageEOFException Class Reference

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2476) or **BytesMessage** (p. 759) is being read.

`#include <src/main/cms/MessageEOFException.h>`Inheritance diagram for cms::MessageEOFException:

Public Member Functions

- **MessageEOFException** () throw ()
- **MessageEOFException** (const **MessageEOFException** &ex) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause) throw ()
- **MessageEOFException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageEOFException** () throw ()

6.365.1 Detailed Description

This exception must be thrown when an unexpected end of stream has been reached when a **StreamMessage** (p. 2476) or **BytesMessage** (p. 759) is being read.

Since:

1.3

6.365.2 Constructor & Destructor Documentation

6.365.2.1 cms::MessageEOFException::MessageEOFException () throw ()

6.365.2.2 cms::MessageEOFException::MessageEOFException (const MessageEOFException & *ex*) throw ()

6.365.2.3 cms::MessageEOFException::MessageEOFException (const std::string & *message*, const std::exception * *cause*) throw ()

6.365.2.4 cms::MessageEOFException::MessageEOFException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.365.2.5 virtual cms::MessageEOFException::~MessageEOFException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageEOFException.h**

6.366 cms::MessageFormatException Class Reference

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

`#include <src/main/cms/MessageFormatException.h>`Inheritance diagram for cms::MessageFormatException:

Public Member Functions

- **MessageFormatException** () throw ()
- **MessageFormatException** (const **MessageFormatException** &ex) throw ()
- **MessageFormatException** (const std::string &message, const std::exception *cause) throw ()
- **MessageFormatException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageFormatException** () throw ()

6.366.1 Detailed Description

This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type. It must also be thrown when equivalent type errors are made with message property values. For example, this exception must be thrown if **StreamMessage.readShort** (p. 2482) is used to read a boolean value.

Since:

1.3

6.366.2 Constructor & Destructor Documentation

6.366.2.1 cms::MessageFormatException::MessageFormatException () throw ()

6.366.2.2 cms::MessageFormatException::MessageFormatException (const MessageFormatException & *ex*) throw ()

6.366.2.3 cms::MessageFormatException::MessageFormatException (const std::string & *message*, const std::exception * *cause*) throw ()

6.366.2.4 cms::MessageFormatException::MessageFormatException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.366.2.5 virtual cms::MessageFormatException::~MessageFormatException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageFormatException.h**

6.367 activemq::commands::MessageId Class Reference

`#include <src/main/activemq/commands/MessageId.h>`Inheritance diagram for activemq::commands::MessageId:

Public Types

- typedef **decaf::lang::PointerComparator< MessageId > COMPARATOR**

Public Member Functions

- **MessageId** ()
- **MessageId** (const **MessageId** &other)
- virtual ~**MessageId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **MessageId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer< ProducerId >** & **getProducerId** () const
- virtual **Pointer< ProducerId >** & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer< ProducerId >** &**producerId**)
- virtual long long **getProducerSequenceId** () const
- virtual void **setProducerSequenceId** (long long **producerSequenceId**)
- virtual long long **getBrokerSequenceId** () const
- virtual void **setBrokerSequenceId** (long long **brokerSequenceId**)
- virtual int **compareTo** (const **MessageId** &value) const
- virtual bool **equals** (const **MessageId** &value) const
- virtual bool **operator==** (const **MessageId** &value) const
- virtual bool **operator<** (const **MessageId** &value) const
- **MessageId** & **operator=** (const **MessageId** &other)

Static Public Attributes

- static const unsigned char **ID_MESSAGEID** = 110

Protected Attributes

- **Pointer< ProducerId > producerId**
- long long **producerSequenceId**
- long long **brokerSequenceId**

6.367.1 Member Typedef Documentation

6.367.1.1 typedef decaf::lang::PointerComparator<MessageId> activemq::commands::MessageId::COMPARATOR

6.367.2 Constructor & Destructor Documentation

6.367.2.1 activemq::commands::MessageId::MessageId ()

6.367.2.2 activemq::commands::MessageId::MessageId (const MessageId & *other*)

6.367.2.3 virtual activemq::commands::MessageId::~MessageId () [virtual]

6.367.3 Member Function Documentation

6.367.3.1 virtual MessageId* activemq::commands::MessageId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.367.3.2 virtual int activemq::commands::MessageId::compareTo (const MessageId & *value*) const [virtual]

6.367.3.3 virtual void activemq::commands::MessageId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.367.3.4 virtual bool activemq::commands::MessageId::equals (const MessageId & *value*) const [virtual]

6.367.3.5 virtual bool activemq::commands::MessageId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are

the same.

Returns:

true if DataStructure's are Equal.

6.367.3.6 virtual long long activemq::commands::MessageId::getBrokerSequenceId () const [virtual]

6.367.3.7 virtual unsigned char activemq::commands::MessageId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.367.3.8 virtual Pointer<ProducerId>& activemq::commands::MessageId::getProducerId ()

[virtual]

6.367.3.9 virtual const Pointer<ProducerId>& activemq::commands::MessageId::getProducerId () const

[virtual]

6.367.3.10 virtual long long activemq::commands::MessageId::getProducerSequenceId () const

[virtual]

6.367.3.11 virtual bool activemq::commands::MessageId::operator< (const MessageId & *value*) const

[virtual]

6.367.3.12 MessageId& activemq::commands::MessageId::operator= (const MessageId & *other*)

6.367.3.13 virtual bool activemq::commands::MessageId::operator== (const MessageId & *value*) const

[virtual]

6.367.3.14 virtual void activemq::commands::MessageId::setBrokerSequenceId (long long *brokerSequenceId*)

[virtual]

6.367.3.15 virtual void activemq::commands::MessageId::setProducerId (const Pointer< ProducerId > & *producerId*)

[virtual]

6.367.3.16 virtual void activemq::commands::MessageId::setProducerSequenceId (long long *producerSequenceId*)

[virtual]

6.367.3.17 virtual std::string activemq::commands::MessageId::toString () const

[virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.367.4 Field Documentation

6.367.4.1 long long activemq::commands::MessageId::brokerSequenceId [protected]

6.367.4.2 const unsigned char activemq::commands::MessageId::ID_MESSAGEID = 110 [static]

6.367.4.3 Pointer<ProducerId> activemq::commands::MessageId::producerId [protected]

6.367.4.4 long long activemq::commands::MessageId::producerSequenceId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessageId.h**

6.368 activemq::wireformat::openwire::marshal::v1::MessageIdMarshalle Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1848).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual ~**MessageIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.368.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1848). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.368.2 Constructor & Destructor Documentation

6.368.2.1 activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::MessageIdMarshaller () [inline]

6.368.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::~MessageIdMarshalle () [inline, virtual]

6.368.3 Member Function Documentation

6.368.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.368.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.368.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.368.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.368.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.368.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.368.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageIdMarshaller.h**

6.369 activemq::wireformat::openwire::marshal::v3::MessageIdMarshalle Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1852).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual ~**MessageIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.369.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1852). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.369.2 Constructor & Destructor Documentation

6.369.2.1 activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::MessageIdMarshaller () [inline]

6.369.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::~MessageIdMarshalle () [inline, virtual]

6.369.3 Member Function Documentation

6.369.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.369.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.369.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.369.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.369.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.369.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.369.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageIdMarshaller.h**

6.370 activemq::wireformat::openwire::marshal::v2::MessageIdMarshalle Class Reference

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1856).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller:

Public Member Functions

- **MessageIdMarshaller** ()
- virtual ~**MessageIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.370.1 Detailed Description

Marshaling code for Open Wire Format for **MessageIdMarshaller** (p. 1856). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.370.2 Constructor & Destructor Documentation

6.370.2.1 activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::MessageIdMarshaller () [inline]

6.370.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::~MessageIdMarshalle () [inline, virtual]

6.370.3 Member Function Documentation

6.370.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.370.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.370.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.370.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.370.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.370.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.370.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageIdMarshaller.h**

6.371 cms::MessageListener Class Reference

A `MessageListener` (p. 1860) object is used to receive asynchronously delivered messages.

`#include <src/main/cms/MessageListener.h>`

Public Member Functions

- virtual ~**MessageListener** ()
- virtual void **onMessage** (const **Message** *message)=0

  Called asynchronously when a new message is received, the message reference can be to any of the ***Message*** *(p. 1753) types.*

6.371.1 Detailed Description

A `MessageListener` (p. 1860) object is used to receive asynchronously delivered messages.

Since:

1.0

6.371.2 Constructor & Destructor Documentation

6.371.2.1 virtual cms::MessageListener::~MessageListener () [inline, virtual]

6.371.3 Member Function Documentation

6.371.3.1 virtual void cms::MessageListener::onMessage (const Message * *message*) [pure virtual]

Called asynchronously when a new message is received, the message reference can be to any of the **Message** (p. 1753) types. a dynamic cast is used to find out what type of message this is. The lifetime of this object is only guaranteed to be for life of the onMessage function after this call-back returns the message may no longer exists. Users should copy the data or clone the message if they wish to retain information that was contained in this **Message** (p. 1753).

It is considered a programming error for this method to throw an exception.

Parameters:

message **Message** (p. 1753) object {const} pointer recipient does not own.

The documentation for this class was generated from the following file:

- src/main/cms/**MessageListener.h**

6.372 activemq::wireformat::openwire::marshal::v2::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1861).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual ~**MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.372.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1861). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.372.2 Constructor & Destructor Documentation

6.372.2.1 activemq::wireformat::openwire::marshal::v2::MessageMarshaller::MessageMarshaller () [inline]

6.372.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessageMarshaller::~MessageMarshaller () [inline, virtual]

6.372.3 Member Function Documentation

6.372.3.1 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 160), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 187), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 276), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 291), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 322), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 396), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 462).

6.372.3.2 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 277), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 397), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 463).

6.372.3.3 virtual int activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 161), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 188), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 277), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 292), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 323), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 397), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 463).

6.372.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 162), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 278), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 324), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 398), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 464).

6.372.3.5 virtual void activemq::wireformat::openwire::marshal::v2::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller** (p. 162), **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller** (p. 189), **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller** (p. 278), **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller** (p. 293), **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller** (p. 324), **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller** (p. 398), and **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller** (p. 464).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessageMarshaller.h**

6.373 activemq::wireformat::openwire::marshal::v3::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1866).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual ~**MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.373.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1866). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.373.2 Constructor & Destructor Documentation

6.373.2.1 activemq::wireformat::openwire::marshal::v3::MessageMarshaller::MessageMarshaller () [inline]

6.373.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessageMarshaller::~MessageMarshaller () [inline, virtual]

6.373.3 Member Function Documentation

6.373.3.1 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 152), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 179), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 268), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 283), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 314), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 388), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 454).

6.373.3.2 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 153), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 269), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 284), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 315), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 389), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 455).

6.373.3.3 virtual int activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 153), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 180), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 269), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 284), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 315), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 389), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 455).

6.373.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 154), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 270), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 316), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 390), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 456).

6.373.3.5 virtual void activemq::wireformat::openwire::marshal::v3::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller** (p. 154), **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller** (p. 181), **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller** (p. 270), **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller** (p. 285), **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller** (p. 316), **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller** (p. 390), and **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller** (p. 456).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessageMarshaller.h**

6.374 activemq::wireformat::openwire::marshal::v1::MessageMarshaller Class Reference

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1871).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessageMarshaller:

Public Member Functions

- **MessageMarshaller** ()
- virtual ~**MessageMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.374.1 Detailed Description

Marshaling code for Open Wire Format for **MessageMarshaller** (p. 1871). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.374.2 Constructor & Destructor Documentation

6.374.2.1 activemq::wireformat::openwire::marshal::v1::MessageMarshaller::MessageMarshaller () [inline]

6.374.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessageMarshaller::~MessageMarshaller () [inline, virtual]

6.374.3 Member Function Documentation

6.374.3.1 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 156), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 183), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 272), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 287), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 318), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 392), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 458).

6.374.3.2 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 393), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 459).

6.374.3.3 virtual int activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 157), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 184), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 273), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 288), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 319), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 393), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 459).

6.374.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 460).

6.374.3.5 virtual void activemq::wireformat::openwire::marshal::v1::MessageMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller** (p. 158), **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller** (p. 185), **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller** (p. 274), **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller** (p. 289), **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller** (p. 320), **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller** (p. 394), and **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller** (p. 460).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessageMarshaller.h**

6.375 cms::MessageNotReadableException Class Reference

This exception must be thrown when a CMS client attempts to read a write-only message.

`#include <src/main/cms/MessageNotReadableException.h>`Inheritance diagram for cms::MessageNotReadableException:

Public Member Functions

- **MessageNotReadableException** () throw ()
- **MessageNotReadableException** (const **MessageNotReadableException** &ex) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception *cause) throw ()
- **MessageNotReadableException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotReadableException** () throw ()

6.375.1 Detailed Description

This exception must be thrown when a CMS client attempts to read a write-only message.

Since:

1.3

6.375.2 Constructor & Destructor Documentation

6.375.2.1 cms::MessageNotReadableException::MessageNotReadableException () throw ()

6.375.2.2 cms::MessageNotReadableException::MessageNotReadableException (const MessageNotReadableException & *ex*) throw ()

6.375.2.3 cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message*, const std::exception * *cause*) throw ()

6.375.2.4 cms::MessageNotReadableException::MessageNotReadableException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.375.2.5 virtual cms::MessageNotReadableException::~MessageNotReadableException () throw () `[virtual]`

The documentation for this class was generated from the following file:

- src/main/cms/**MessageNotReadableException.h**

6.376 cms::MessageNotWriteableException Class Reference

This exception must be thrown when a CMS client attempts to write to a read-only message.

`#include <src/main/cms/MessageNotWriteableException.h>`Inheritance diagram for cms::MessageNotWriteableException:

Public Member Functions

- **MessageNotWriteableException** () throw ()
- **MessageNotWriteableException** (const **MessageNotWriteableException** &ex) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause) throw ()
- **MessageNotWriteableException** (const std::string &message, const std::exception *cause, const std::vector< std::pair< std::string, int > > &stackTrace) throw ()
- virtual ~**MessageNotWriteableException** () throw ()

6.376.1 Detailed Description

This exception must be thrown when a CMS client attempts to write to a read-only message.

Since:

1.3

6.376.2 Constructor & Destructor Documentation

6.376.2.1 cms::MessageNotWriteableException::MessageNotWriteableException () throw ()

6.376.2.2 cms::MessageNotWriteableException::MessageNotWriteableException (const MessageNotWriteableException & *ex*) throw ()

6.376.2.3 cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message*, const std::exception * *cause*) throw ()

6.376.2.4 cms::MessageNotWriteableException::MessageNotWriteableException (const std::string & *message*, const std::exception * *cause*, const std::vector< std::pair< std::string, int > > & *stackTrace*) throw ()

6.376.2.5 virtual cms::MessageNotWriteableException::~MessageNotWriteableException () throw () [virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**MessageNotWriteableException.h**

6.377 cms::MessageProducer Class Reference

A client uses a `MessageProducer` (p. 1878) object to send messages to a **Destination** (p. 1190).

`#include <src/main/cms/MessageProducer.h>`Inheritance diagram for cms::MessageProducer:

Public Member Functions

- virtual ~**MessageProducer** ()
- virtual void **send** (**Message** *message)=0 throw (CMSException)

  Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (**Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (CMSException)

  Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **Destination** *destination, **Message** *message)=0 throw (CMSException)

  Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **send** (const **Destination** *destination, **Message** *message, int deliveryMode, int priority, long long timeToLive)=0 throw (CMSException)

  Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

- virtual void **setDeliveryMode** (int mode)=0 throw (CMSException)

  Sets the delivery mode for this Producer.

- virtual int **getDeliveryMode** () const =0 throw (CMSException)

  Gets the delivery mode for this Producer.

- virtual void **setDisableMessageID** (bool value)=0 throw (CMSException)

  *Sets if **Message** (p. 1753) Ids are disabled for this Producer.*

- virtual bool **getDisableMessageID** () const =0 throw (CMSException)

  *Gets if **Message** (p. 1753) Ids are disabled for this Producer.*

- virtual void **setDisableMessageTimeStamp** (bool value)=0 throw (CMSException)

  *Sets if **Message** (p. 1753) Time Stamps are disabled for this Producer.*

- virtual bool **getDisableMessageTimeStamp** () const =0 throw (CMSException)

  *Gets if **Message** (p. 1753) Time Stamps are disabled for this Producer.*

- virtual void **setPriority** (int priority)=0 throw (CMSException)

  Sets the Priority that this Producers sends messages at.

- virtual int **getPriority** () const =0 throw (CMSException)
  Gets the Priority level that this producer sends messages at.
- virtual void **setTimeToLive** (long long time)=0 throw (CMSException)
  Sets the Time to Live that this Producers sends messages with.
- virtual long long **getTimeToLive** () const =0 throw (CMSException)
  Gets the Time to Live that this producer sends messages with.

6.377.1 Detailed Description

A client uses a `MessageProducer (p. 1878)` object to send messages to a **Destination** (p. 1190). A `MessageProducer (p. 1878)` object is created by passing a `Destination (p. 1190)` object to a message-producer creation method supplied by a session.

A client also has the option of creating a message producer without supplying a destination. In this case, a **Destination** (p. 1190) must be provided with every send operation. A typical use for this kind of message producer is to send replies to requests using the request's CMSReplyTo destination.

A client can specify a default delivery mode, priority, and time to live for messages sent by a message producer. It can also specify the delivery mode, priority, and time to live for an individual message.

A client can specify a time-to-live value in milliseconds for each message it sends. This value defines a message expiration time that is the sum of the message's time-to-live and the GMT when it is sent (for transacted sends, this is the time the client sends the message, not the time the transaction is committed).

Since:

1.0

6.377.2 Constructor & Destructor Documentation

6.377.2.1 virtual cms::MessageProducer::~MessageProducer () `[inline, virtual]`

6.377.3 Member Function Documentation

6.377.3.1 virtual int cms::MessageProducer::getDeliveryMode () const throw (CMSException) `[pure virtual]`

Gets the delivery mode for this Producer.

Returns:

The **DeliveryMode** (p. 1188)

Exceptions:

CMSException *(p. **850**)* - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 778), and **activemq::core::ActiveMQProducer** (p. 327).

6.377.3.2 virtual bool cms::MessageProducer::getDisableMessageID () const throw (CMSException) [pure virtual]

Gets if **Message** (p. 1753) Ids are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 779), and **activemq::core::ActiveMQProducer** (p. 327).

6.377.3.3 virtual bool cms::MessageProducer::getDisableMessageTimeStamp () const throw (CMSException) [pure virtual]

Gets if **Message** (p. 1753) Time Stamps are disabled for this Producer.

Returns:

boolean indicating enable / disable (true / false)

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 779), and **activemq::core::ActiveMQProducer** (p. 327).

6.377.3.4 virtual int cms::MessageProducer::getPriority () const throw (CMSException) [pure virtual]

Gets the Priority level that this producer sends messages at.

Returns:

int based priority level

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 779), and **activemq::core::ActiveMQProducer** (p. 328).

6.377.3.5 virtual long long cms::MessageProducer::getTimeToLive () const throw (CMSException) [pure virtual]

Gets the Time to Live that this producer sends messages with.

Returns:

Time to live value in milliseconds

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 779), and **activemq::core::ActiveMQProducer** (p. 328).

6.377.3.6 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (CMSException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

destination The destination on which to send the message

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 780), and **activemq::core::ActiveMQProducer** (p. 329).

6.377.3.7 virtual void cms::MessageProducer::send (const Destination * *destination*, Message * *message*) throw (CMSException) [pure virtual]

Sends the message to the designated destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

destination The destination on which to send the message

message the message to be sent.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 780), and **activemq::core::ActiveMQProducer** (p. 329).

6.377.3.8 virtual void cms::MessageProducer::send (Message * *message*, int *deliveryMode*, int *priority*, long long *timeToLive*) throw (CMSException) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it.

Parameters:

message The message to be sent.

deliveryMode The delivery mode to be used.

priority The priority for this message.

timeToLive The time to live value for this message in milliseconds.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 780), and **activemq::core::ActiveMQProducer** (p. 330).

6.377.3.9 virtual void cms::MessageProducer::send (Message * *message*) throw (CMSException) [pure virtual]

Sends the message to the default producer destination, but does not take ownership of the message, caller must still destroy it. Uses default values for deliveryMode, priority, and time to live.

Parameters:

message The message to be sent.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 781), and **activemq::core::ActiveMQProducer** (p. 330).

6.377.3.10 virtual void cms::MessageProducer::setDeliveryMode (int *mode*) throw (CMSException) [pure virtual]

Sets the delivery mode for this Producer.

Parameters:

mode The **DeliveryMode** (p. 1188)

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 781), and **activemq::core::ActiveMQProducer** (p. 330).

6.377.3.11 virtual void cms::MessageProducer::setDisableMessageID (bool *value*) throw (CMSException) [pure virtual]

Sets if **Message** (p. 1753) Ids are disabled for this Producer.

Parameters:

value boolean indicating enable / disable (true / false)

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 781), and **activemq::core::ActiveMQProducer** (p. 330).

6.377.3.12 virtual void cms::MessageProducer::setDisableMessageTimeStamp (bool *value*) throw (CMSException) [pure virtual]

Sets if **Message** (p. 1753) Time Stamps are disabled for this Producer.

Parameters:

value - boolean indicating enable / disable (true / false)

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 782), and **activemq::core::ActiveMQProducer** (p. 331).

6.377.3.13 virtual void cms::MessageProducer::setPriority (int *priority*) throw (CMSException) [pure virtual]

Sets the Priority that this Producers sends messages at.

Parameters:

priority int value for Priority level

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 782), and **activemq::core::ActiveMQProducer** (p. 331).

6.377.3.14 virtual void cms::MessageProducer::setTimeToLive (long long *time*) throw (CMSException) [pure virtual]

Sets the Time to Live that this Producers sends messages with. This value will be used if the time to live is not specified via the send method.

Parameters:

time default time to live value in milliseconds

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::cmsutil::CachedProducer** (p. 782), and **activemq::core::ActiveMQProducer** (p. 331).

The documentation for this class was generated from the following file:

- src/main/cms/**MessageProducer.h**

6.378 activemq::wireformat::openwire::utils::MessagePropertyIntercepto Class Reference

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

`#include <src/main/activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>`

Public Member Functions

- **MessagePropertyInterceptor** (**commands::Message** *message, **util::PrimitiveMap** *properties) throw (decaf::lang::exceptions::NullPointerException)

  Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

- virtual ~**MessagePropertyInterceptor** ()
- virtual bool **getBooleanProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a boolean property.

- virtual unsigned char **getByteProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a byte property.

- virtual double **getDoubleProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a double property.

- virtual float **getFloatProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a float property.

- virtual int **getIntProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a int property.

- virtual long long **getLongProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a long property.

- virtual short **getShortProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a short property.

- virtual std::string **getStringProperty** (const std::string &name) const throw (exceptions::ActiveMQException)

  Gets a string property.

- virtual void **setBooleanProperty** (const std::string &name, bool value) throw (exceptions::ActiveMQException)

Sets a boolean property.

- virtual void **setByteProperty** (const std::string &name, unsigned char value) throw (exceptions::ActiveMQException)

  Sets a byte property.

- virtual void **setDoubleProperty** (const std::string &name, double value) throw (exceptions::ActiveMQException)

  Sets a double property.

- virtual void **setFloatProperty** (const std::string &name, float value) throw (exceptions::ActiveMQException)

  Sets a float property.

- virtual void **setIntProperty** (const std::string &name, int value) throw (exceptions::ActiveMQException)

  Sets a int property.

- virtual void **setLongProperty** (const std::string &name, long long value) throw (exceptions::ActiveMQException)

  Sets a long property.

- virtual void **setShortProperty** (const std::string &name, short value) throw (exceptions::ActiveMQException)

  Sets a short property.

- virtual void **setStringProperty** (const std::string &name, const std::string &value) throw (exceptions::ActiveMQException)

  Sets a string property.

6.378.1 Detailed Description

Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties. Currently the only properties that are intercepted and handled are:

Name | Conversion Supported ---------- JMSXDeliveryCount | Int, Long, String JMSXGroupID | String JMSXGroupSeq | Int, Long, String

6.378.2 Constructor & Destructor Documentation

6.378.2.1 activemq::wireformat::openwire::utils::MessagePropertyInterceptor::MessagePropertyInte (commands::Message * *message*, util::PrimitiveMap * *properties*) throw (decaf::lang::exceptions::NullPointerException)

Constructor, accepts the Message that will be used to store JMS reserved property values, and the PrimitiveMap to get and set the rest to.

Parameters:

message - The Message to store reserved property data in

properties - The PrimitiveMap to store the rest of the properties in.

Exceptions:

NullPointerException if either param is NULL

6.378.2.2 virtual activemq::wireformat::openwire::utils::MessagePropertyInterceptor::~MessagePropertyInt () [virtual]

6.378.3 Member Function Documentation

6.378.3.1 virtual bool activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getBooleanProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a boolean property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.2 virtual unsigned char activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getByteProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a byte property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.3 virtual double activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getDoubleProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a double property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.4 virtual float activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getFloatProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a float property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.5 virtual int activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getIntProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a int property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.6 virtual long long activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getLongProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a long property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.7 virtual short activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getShortProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a short property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.8 virtual std::string activemq::wireformat::openwire::utils::MessagePropertyInterceptor::getStringProperty (const std::string & *name*) const throw (exceptions::ActiveMQException) [virtual]

Gets a string property.

Parameters:

name The name of the property to retrieve.

Returns:

The value for the named property.

Exceptions:

CMSException if the property does not exist.

6.378.3.9 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setBooleanProperty (const std::string & *name*, bool *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a boolean property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.10 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setByteProperty (const std::string & *name*, unsigned char *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a byte property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.11 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setDoubleProperty (const std::string & *name*, double *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a double property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.12 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setFloatProperty (const std::string & *name*, float *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a float property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.13 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setIntProperty (const std::string & *name*, int *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a int property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.14 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setLongProperty (const std::string & *name*, long long *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a long property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.15 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setShortProperty (const std::string & *name*, short *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a short property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

6.378.3.16 virtual void activemq::wireformat::openwire::utils::MessagePropertyInterceptor::setStringProperty (const std::string & *name*, const std::string & *value*) throw (exceptions::ActiveMQException) [virtual]

Sets a string property.

Parameters:

name The name of the property to retrieve.

value The value for the named property.

Exceptions:

CMSException

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**MessagePropertyInterceptor.h**

6.379 activemq::commands::MessagePull Class Reference

`#include <src/main/activemq/commands/MessagePull.h>`Inheritance diagram for activemq::commands::MessagePull:

Public Member Functions

- **MessagePull** ()
- virtual ~**MessagePull** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **MessagePull** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConsumerId** > & **getConsumerId** () const
- virtual **Pointer**< **ConsumerId** > & **getConsumerId** ()
- virtual void **setConsumerId** (const **Pointer**< **ConsumerId** > &**consumerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual long long **getTimeout** () const
- virtual void **setTimeout** (long long **timeout**)
- virtual const std::string & **getCorrelationId** () const
- virtual std::string & **getCorrelationId** ()
- virtual void **setCorrelationId** (const std::string &**correlationId**)
- virtual const **Pointer**< **MessageId** > & **getMessageId** () const
- virtual **Pointer**< **MessageId** > & **getMessageId** ()
- virtual void **setMessageId** (const **Pointer**< **MessageId** > &**messageId**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_MESSAGEPULL** = 20

Protected Member Functions

- **MessagePull** (const **MessagePull** &)
- **MessagePull** & **operator**= (const **MessagePull** &)

Protected Attributes

- **Pointer**< **ConsumerId** > **consumerId**
- **Pointer**< **ActiveMQDestination** > **destination**
- long long **timeout**
- std::string **correlationId**
- **Pointer**< **MessageId** > **messageId**

6.379.1 Constructor & Destructor Documentation

6.379.1.1 activemq::commands::MessagePull::MessagePull (const MessagePull &) [inline, protected]

6.379.1.2 activemq::commands::MessagePull::MessagePull ()

6.379.1.3 virtual activemq::commands::MessagePull::~MessagePull () [virtual]

6.379.2 Member Function Documentation

6.379.2.1 virtual MessagePull* activemq::commands::MessagePull::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.379.2.2 virtual void activemq::commands::MessagePull::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.379.2.3 virtual bool activemq::commands::MessagePull::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.379.2.4 virtual Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () [virtual]

6.379.2.5 virtual const Pointer<ConsumerId>& activemq::commands::MessagePull::getConsumerId () const [virtual]

6.379.2.6 virtual std::string& activemq::commands::MessagePull::getCorrelationId () [virtual]

6.379.2.7 virtual const std::string& activemq::commands::MessagePull::getCorrelationId () const [virtual]

6.379.2.8 virtual unsigned char activemq::commands::MessagePull::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.379.2.9 virtual Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () [virtual]

6.379.2.10 virtual const Pointer<ActiveMQDestination>& activemq::commands::MessagePull::getDestination () const [virtual]

6.379.2.11 virtual Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () [virtual]

6.379.2.12 virtual const Pointer<MessageId>& activemq::commands::MessagePull::getMessageId () const [virtual]

6.379.2.13 virtual long long activemq::commands::MessagePull::getTimeout () const [virtual]

6.379.2.14 MessagePull& activemq::commands::MessagePull::operator= (const MessagePull &) [inline, protected]

6.379.2.15 virtual void activemq::commands::MessagePull::setConsumerId (const Pointer< ConsumerId > & *consumerId*) [virtual]

6.379.2.16 virtual void activemq::commands::MessagePull::setCorrelationId (const std::string & *correlationId*) [virtual]

6.379.2.17 virtual void activemq::commands::MessagePull::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.379.2.18 virtual void activemq::commands::MessagePull::setMessageId (const Pointer< MessageId > & *messageId*) [virtual]

6.379.2.19 virtual void activemq::commands::MessagePull::setTimeout (long long *timeout*) [virtual]

6.379.2.20 virtual std::string activemq::commands::MessagePull::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.379.2.21 virtual Pointer<Command> activemq::commands::MessagePull::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the

visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.379.3 Field Documentation

6.379.3.1 Pointer<ConsumerId> activemq::commands::MessagePull::consumerId [protected]

6.379.3.2 std::string activemq::commands::MessagePull::correlationId [protected]

6.379.3.3 Pointer<ActiveMQDestination> activemq::commands::MessagePull::destination [protected]

6.379.3.4 const unsigned char activemq::commands::MessagePull::ID_MESSAGEPULL = 20 [static]

6.379.3.5 Pointer<MessageId> activemq::commands::MessagePull::messageId [protected]

6.379.3.6 long long activemq::commands::MessagePull::timeout [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**MessagePull.h**

6.380 activemq::wireformat::openwire::marshal::v2::MessagePullMarshal Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1898).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/MessagePullMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual ~**MessagePullMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.380.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1898). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.380.2 Constructor & Destructor Documentation

6.380.2.1 activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::MessagePullMarsha () [inline]

6.380.2.2 virtual activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::~MessagePullMarsh () [inline, virtual]

6.380.3 Member Function Documentation

6.380.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.380.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.380.3.3 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.380.3.4 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.380.3.5 virtual int activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.380.3.6 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.380.3.7 virtual void activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**MessagePullMarshaller.h**

6.381 activemq::wireformat::openwire::marshal::v3::MessagePullMarshal Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1902).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/MessagePullMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual ~**MessagePullMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.381.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1902). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.381.2 Constructor & Destructor Documentation

6.381.2.1 activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::MessagePullMarsha () [inline]

6.381.2.2 virtual activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::~MessagePullMarsh () [inline, virtual]

6.381.3 Member Function Documentation

6.381.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.381.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.381.3.3 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.381.3.4 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.381.3.5 virtual int activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.381.3.6 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.381.3.7 virtual void activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**MessagePullMarshaller.h**

6.382 activemq::wireformat::openwire::marshal::v1::MessagePullMarshal Class Reference

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1906).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/MessagePullMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller:

Public Member Functions

- **MessagePullMarshaller** ()
- virtual ~**MessagePullMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.382.1 Detailed Description

Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1906). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.382.2 Constructor & Destructor Documentation

6.382.2.1 activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::MessagePullMarsha () [inline]

6.382.2.2 virtual activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::~MessagePullMarsh () [inline, virtual]

6.382.3 Member Function Documentation

6.382.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.382.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.382.3.3 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.382.3.4 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.382.3.5 virtual int activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.382.3.6 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.382.3.7 virtual void activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**MessagePullMarshaller.h**

6.383 activemq::transport::mock::MockTransport Class Reference

The **MockTransport** (p. 1910) defines a base level **Transport** (p. 2608) class that is intended to be used in place of an a regular protocol **Transport** (p. 2608) suck as TCP.

`#include <src/main/activemq/transport/mock/MockTransport.h>`Inheritance diagram for activemq::transport::mock::MockTransport:

Public Member Functions

- **MockTransport** (const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **Pointer**< **ResponseBuilder** > &responseBuilder)
- virtual ~**MockTransport** ()
- void **setResponseBuilder** (const **Pointer**< **ResponseBuilder** > &responseBuilder)

  *Sets the **ResponseBuilder** (p. 2235) that this class uses to create Responses to Commands sent.*

- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual void **setOutgoingListener** (**TransportListener** *listener)

  *Sets a Listener that gets notified for every command that would have been sent by this **transport** (p. 67) to the Broker, this allows a client to verify that its messages are making it to the wire.*

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat AMQCPP_UNUSED)

  Sets the WireFormat instance to use.

- virtual void **setTransportListener** (**TransportListener** *listener)

  *Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*

- virtual **TransportListener** * **getTransportListener** () const

  *Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*

- virtual void **fireCommand** (const **Pointer**< **Command** > &command)

  *Fires a Command back through this **transport** (p. 67) to its registered CommandListener if there is one.*

- virtual void **fireException** (const **exceptions::ActiveMQException** &ex)

*Fires a Exception back through this **transport** (p. 67) to its registered ExceptionListener if there is one.*

- virtual void **start** () throw (cms::CMSException)

  Starts the service.

- virtual void **close** () throw (cms::CMSException)

  Closes this object and deallocates the appropriate resources.

- virtual **Transport** * **narrow** (const std::type_info &typeId)

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri AMQCPP_UNUSED) throw (decaf::io::IOException)

  reconnect to another location

- bool **isFailOnSendMessage** () const
- void **setFailOnSendMessage** (bool value)
- int **getNumSentMessageBeforeFail** () const
- void **setNumSentMessageBeforeFail** (int value)
- int **getNumSentMessages** () const
- void **setNumSentMessages** (int value)
- bool **isFailOnReceiveMessage** () const
- void **setFailOnReceiveMessage** (bool value)
- int **getNumReceivedMessageBeforeFail** () const
- void **setNumReceivedMessageBeforeFail** (int value)
- int **getNumReceivedMessages** () const
- void **setNumReceivedMessages** (int value)

Static Public Member Functions

- static **MockTransport** * **getInstance** ()

6.383.1 Detailed Description

The **MockTransport** (p. 1910) defines a base level **Transport** (p. 2608) class that is intended to be used in place of an a regular protocol **Transport** (p. 2608) suck as TCP. This **Transport** (p. 2608) assumes that it is the base **Transport** (p. 2608) in the Transports stack, and destroys any Transports that are passed to it in its constructor.

This **Transport** (p. 2608) defines an Interface **ResponseBuilder** (p. 2235) which must be implemented by any protocol for which the **Transport** (p. 2608) is used to Emulate. The **Transport** (p. 2608) hands off all outbound **commands** (p. 59) to the **ResponseBuilder** (p. 2235) for processing, it is up to the builder to create appropriate responses and schedule any asynchronous messages that might result from a message sent to the Broker.

6.383.2 Constructor & Destructor Documentation

6.383.2.1 activemq::transport::mock::MockTransport::MockTransport (const Pointer< wireformat::WireFormat > & *wireFormat*, const Pointer< ResponseBuilder > & *responseBuilder*)

6.383.2.2 virtual activemq::transport::mock::MockTransport::~MockTransport () [inline, virtual]

6.383.3 Member Function Documentation

6.383.3.1 virtual void activemq::transport::mock::MockTransport::close () throw (cms::CMSException) [inline, virtual]

Closes this object and deallocates the appropriate resources. The object is generally no longer usable after calling close.

Exceptions:

CMSException - If an error occurs while the resource is being closed.

Implements **cms::Closeable** (p. 838).

6.383.3.2 virtual void activemq::transport::mock::MockTransport::fireCommand (const Pointer< Command > & *command*) [inline, virtual]

Fires a Command back through this **transport** (p. 67) to its registered CommandListener if there is one.

Parameters:

command - Command to send to the Listener.

6.383.3.3 virtual void activemq::transport::mock::MockTransport::fireException (const exceptions::ActiveMQException & *ex*) [inline, virtual]

Fires a Exception back through this **transport** (p. 67) to its registered ExceptionListener if there is one.

Parameters:

ex The Exception that will be passed on the the **Transport** (p. 2608) listener.

6.383.3.4 static MockTransport* activemq::transport::mock::MockTransport::getInstance () [inline, static]

6.383.3.5 int activemq::transport::mock::MockTransport::getNumReceivedMessageBeforeFail () const [inline]

6.383.3.6 int activemq::transport::mock::MockTransport::getNumReceivedMessages () const [inline]

6.383.3.7 int activemq::transport::mock::MockTransport::getNumSentMessageBeforeFail () const [inline]

6.383.3.8 int activemq::transport::mock::MockTransport::getNumSentMessages () const [inline]

6.383.3.9 virtual std::string activemq::transport::mock::MockTransport::getRemoteAddress () const [inline, virtual]

Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2609).

6.383.3.10 virtual TransportListener* activemq::transport::mock::MockTransport::getTransportListener () const [inline, virtual]

Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Returns:

The listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2609).

6.383.3.11 virtual bool activemq::transport::mock::MockTransport::isClosed () const [inline, virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Implements **activemq::transport::Transport** (p. 2609).

6.383.3.12 virtual bool activemq::transport::mock::MockTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2610).

6.383.3.13 bool activemq::transport::mock::MockTransport::isFailOnReceiveMessage () const [inline]

6.383.3.14 bool activemq::transport::mock::MockTransport::isFailOnSendMessage () const [inline]

6.383.3.15 virtual bool activemq::transport::mock::MockTransport::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2610).

6.383.3.16 virtual Transport* activemq::transport::mock::MockTransport::narrow (const std::type_info & *typeId*) [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2610).

6.383.3.17 virtual void activemq::transport::mock::MockTransport::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2610).

6.383.3.18 virtual void activemq::transport::mock::MockTransport::reconnect (const decaf::net::URI &uri *AMQCPP_UNUSED*) throw (decaf::io::IOException) [inline, virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

6.383.3.19 virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.

timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2611).

6.383.3.20 virtual Pointer<Response> activemq::transport::mock::MockTransport::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2612).

6.383.3.21 void activemq::transport::mock::MockTransport::setFailOnReceiveMessage (bool *value*) [inline]

6.383.3.22 void activemq::transport::mock::MockTransport::setFailOnSendMessage (bool *value*) [inline]

6.383.3.23 void activemq::transport::mock::MockTransport::setNumReceivedMessageBeforeFail (int *value*) [inline]

6.383.3.24 void activemq::transport::mock::MockTransport::setNumReceivedMessages (int *value*) [inline]

6.383.3.25 void activemq::transport::mock::MockTransport::setNumSentMessageBeforeFail (int *value*) [inline]

6.383.3.26 void activemq::transport::mock::MockTransport::setNumSentMessages (int *value*) [inline]

6.383.3.27 virtual void activemq::transport::mock::MockTransport::setOutgoingListener (TransportListener * *listener*) [inline, virtual]

Sets a Listener that gets notified for every command that would have been sent by this **transport** (p. 67) to the Broker, this allows a client to verify that its messages are making it to the wire.

Parameters:

listener - The CommandListener to notify for each message

6.383.3.28 void activemq::transport::mock::MockTransport::setResponseBuilder (const Pointer< ResponseBuilder > & *responseBuilder*) [inline]

Sets the **ResponseBuilder** (p. 2235) that this class uses to create Responses to Commands sent. These are either real Response Objects, or Commands that would have been sent Asynchronously be the Broker.

Parameters:

responseBuilder - The **ResponseBuilder** (p. 2235) to use from now on.

6.383.3.29 virtual void activemq::transport::mock::MockTransport::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Parameters:

listener the listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2612).

6.383.3.30 virtual void activemq::transport::mock::MockTransport::setWireFormat (const Pointer< wireformat::WireFormat > &wireFormat *AMQCPP_UNUSED*) [inline, virtual]

Sets the WireFormat instance to use.

Parameters:

wireFormat WireFormat the object used to encode / decode **commands** (p. 59).

6.383.3.31 virtual void activemq::transport::mock::MockTransport::start () throw (cms::CMSException) [inline, virtual]

Starts the service.

Exceptions:

CMSException if an internal error occurs while starting.

Implements **cms::Startable** (p. 2413).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransport.h**

6.384 activemq::transport::mock::MockTransportFactory Class Reference

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

`#include <src/main/activemq/transport/mock/MockTransportFactory.h>`Inheritance diagram for activemq::transport::mock::MockTransportFactory:

Public Member Functions

- virtual ~**MockTransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

Protected Member Functions

- virtual **Pointer< Transport > doCreateComposite** (const **decaf::net::URI** &location, const **Pointer< wireformat::WireFormat >** &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

6.384.1 Detailed Description

Manufactures MockTransports, which are objects that read from input streams and write to output streams.

6.384.2 Constructor & Destructor Documentation

6.384.2.1 virtual activemq::transport::mock::MockTransportFactory::~MockTransportFactory () [inline, virtual]

6.384.3 Member Function Documentation

6.384.3.1 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::create (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2614).

6.384.3.2 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::createComposite (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2615).

6.384.3.3 virtual Pointer<Transport> activemq::transport::mock::MockTransportFactory::doCreateComposite (const decaf::net::URI & *location*, const Pointer< wireformat::WireFormat > & *wireFormat*, const decaf::util::Properties & *properties*) throw (exceptions::ActiveMQException) [protected, virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 2608).

properties - Properties to apply to the **transport** (p. 67).

Returns:

Pointer to a new **Transport** (p. 2608) instance.

Exceptions:

ActiveMQexception if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**MockTransportFactory.h**

6.385 decaf::util::concurrent::Mutex Class Reference

Creates a pthread_mutex_t object.

`#include <src/main/decaf/util/concurrent/Mutex.h>`Inheritance diagram for decaf::util::concurrent::Mutex:

Public Member Functions

- **Mutex** ()

  Constructor - creates and initializes the mutex.

- virtual ~**Mutex** ()

  Destructor - destroys the mutex object.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.385.1 Detailed Description

Creates a pthread_mutex_t object. The object is created such that successive **locks** (p. 119) from the same thread is allowed and will be successful.

See also:

pthread_mutex_t

6.385.2 Constructor & Destructor Documentation

6.385.2.1 decaf::util::concurrent::Mutex::Mutex ()

Constructor - creates and initializes the mutex.

6.385.2.2 virtual decaf::util::concurrent::Mutex::~Mutex () [virtual]

Destructor - destroys the mutex object.

6.385.3 Member Function Documentation

6.385.3.1 virtual void decaf::util::concurrent::Mutex::lock () throw (lang::Exception) [virtual]

Locks the object.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::lock(), decaf::util::StlMap< std::string, cms::Topic * >::lock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::lock(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::lock().

6.385.3.2 virtual void decaf::util::concurrent::Mutex::notify () throw (lang::Exception) [virtual]

Signals a waiter on this object that it can now wake up and continue.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notify(), decaf::util::StlMap< std::string, cms::Topic * >::notify(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notify(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::notify().

6.385.3.3 virtual void decaf::util::concurrent::Mutex::notifyAll () throw (lang::Exception) [virtual]

Signals the waiters on this object that it can now wake up and continue.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::notifyAll(), decaf::util::StlMap< std::string, cms::Topic * >::notifyAll(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::notifyAll(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::notifyAll().

6.385.3.4 virtual void decaf::util::concurrent::Mutex::unlock () throw (lang::Exception) [virtual]

Unlocks the object.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::unlock(), decaf::util::StlMap< std::string, cms::Topic * >::unlock(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::unlock(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::unlock().

6.385.3.5 virtual void decaf::util::concurrent::Mutex::wait (unsigned long *millisecs*) throw (lang::Exception) [virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in milliseconds to wait.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.385.3.6 virtual void decaf::util::concurrent::Mutex::wait () throw (lang::Exception) [virtual]

Waits on a signal from this object, which is generated by a call to Notify.

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::wait(), decaf::util::StlMap< std::string, cms::Topic * >::wait(), decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >::wait(), and decaf::util::AbstractCollection< Pointer< BackupTransport > >::wait().

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Mutex.h**

6.386 activemq::commands::NetworkBridgeFilter Class Reference

`#include <src/main/activemq/commands/NetworkBridgeFilter.h>`Inheritance diagram for activemq::commands::NetworkBridgeFilter:

Public Member Functions

- **NetworkBridgeFilter** ()
- virtual ~**NetworkBridgeFilter** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **NetworkBridgeFilter** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual int **getNetworkTTL** () const
- virtual void **setNetworkTTL** (int **networkTTL**)
- virtual const **Pointer**< **BrokerId** > & **getNetworkBrokerId** () const
- virtual **Pointer**< **BrokerId** > & **getNetworkBrokerId** ()
- virtual void **setNetworkBrokerId** (const **Pointer**< **BrokerId** > &**networkBrokerId**)

Static Public Attributes

- static const unsigned char **ID_NETWORKBRIDGEFILTER** = 91

Protected Member Functions

- **NetworkBridgeFilter** (const **NetworkBridgeFilter** &)
- **NetworkBridgeFilter** & **operator**= (const **NetworkBridgeFilter** &)

Protected Attributes

- int **networkTTL**
- **Pointer**< **BrokerId** > **networkBrokerId**

6.386.1 Constructor & Destructor Documentation

6.386.1.1 activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter (const NetworkBridgeFilter &) [inline, protected]

6.386.1.2 activemq::commands::NetworkBridgeFilter::NetworkBridgeFilter ()

6.386.1.3 virtual activemq::commands::NetworkBridgeFilter::~NetworkBridgeFilter () [virtual]

6.386.2 Member Function Documentation

6.386.2.1 virtual NetworkBridgeFilter* activemq::commands::NetworkBridgeFilter::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.386.2.2 virtual void activemq::commands::NetworkBridgeFilter::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.386.2.3 virtual bool activemq::commands::NetworkBridgeFilter::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.386.2.4 virtual unsigned char activemq::commands::NetworkBridgeFilter::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.386.2.5 virtual Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () [virtual]

6.386.2.6 virtual const Pointer<BrokerId>& activemq::commands::NetworkBridgeFilter::getNetworkBrokerId () const [virtual]

6.386.2.7 virtual int activemq::commands::NetworkBridgeFilter::getNetworkTTL () const [virtual]

6.386.2.8 NetworkBridgeFilter& activemq::commands::NetworkBridgeFilter::operator= (const NetworkBridgeFilter &) [inline, protected]

6.386.2.9 virtual void activemq::commands::NetworkBridgeFilter::setNetworkBrokerId (const Pointer< BrokerId > & *networkBrokerId*) [virtual]

6.386.2.10 virtual void activemq::commands::NetworkBridgeFilter::setNetworkTTL (int *networkTTL*) [virtual]

6.386.2.11 virtual std::string activemq::commands::NetworkBridgeFilter::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.386.3 Field Documentation

6.386.3.1 const unsigned char activemq::commands::NetworkBridgeFilter::ID_NETWORKBRIDGEFILTER = 91 [static]

6.386.3.2 Pointer<BrokerId> activemq::commands::NetworkBridgeFilter::networkBrokerId [protected]

6.386.3.3 int activemq::commands::NetworkBridgeFilter::networkTTL [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**NetworkBridgeFilter.h**

6.387 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilter Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1927).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/NetworkBridgeFilterMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual ~**NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.387.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1927). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.387.2 Constructor & Destructor Documentation

6.387.2.1 activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::NetworkBri () [inline]

6.387.2.2 virtual activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::~NetworkB () [inline, virtual]

6.387.3 Member Function Documentation

6.387.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.387.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.387.3.3 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.387.3.4 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.387.3.5 virtual int activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.387.3.6 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.387.3.7 virtual void activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**NetworkBridgeFilterMarshaller.h**

6.388 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilter Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1931).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/NetworkBridgeFilterMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual ~**NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.388.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1931). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.388.2 Constructor & Destructor Documentation

6.388.2.1 activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::NetworkBri () [inline]

6.388.2.2 virtual activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::~NetworkB () [inline, virtual]

6.388.3 Member Function Documentation

6.388.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.388.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.388.3.3 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.388.3.4 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.388.3.5 virtual int activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.388.3.6 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.388.3.7 virtual void activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**NetworkBridgeFilterMarshaller.h**

6.389 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilter Class Reference

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1935).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/NetworkBridgeFilterMarshaller.h>`Inheri
diagram for activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller:

Public Member Functions

- **NetworkBridgeFilterMarshaller** ()
- virtual ~**NetworkBridgeFilterMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.389.1 Detailed Description

Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1935). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.389.2 Constructor & Destructor Documentation

6.389.2.1 activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::NetworkBri () [inline]

6.389.2.2 virtual activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::~NetworkB () [inline, virtual]

6.389.3 Member Function Documentation

6.389.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.389.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::getDataStruct () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.389.3.3 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.389.3.4 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::looseUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.389.3.5 virtual int activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.389.3.6 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.389.3.7 virtual void activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller::tightUnmarsh (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**NetworkBridgeFilterMarshaller.h**

6.390 decaf::net::NoRouteToHostException Class Reference

`#include <src/main/decaf/net/NoRouteToHostException.h>`Inheritance diagram for decaf::net::NoRouteToHostException:

Public Member Functions

- **NoRouteToHostException** () throw ()
  Default Constructor.
- **NoRouteToHostException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **NoRouteToHostException** (const **NoRouteToHostException** &ex) throw ()
  Copy Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **NoRouteToHostException** (const std::exception ***cause**) throw ()
  Constructor.
- **NoRouteToHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoRouteToHostException** * **clone** () const
  Clones this exception.
- virtual ~**NoRouteToHostException** () throw ()

6.390.1 Constructor & Destructor Documentation

6.390.1.1 decaf::net::NoRouteToHostException::NoRouteToHostException () throw () [inline]

Default Constructor.

6.390.1.2 decaf::net::NoRouteToHostException::NoRouteToHostException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.390.1.3 decaf::net::NoRouteToHostException::NoRouteToHostException (const NoRouteToHostException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.390.1.4 decaf::net::NoRouteToHostException::NoRouteToHostException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.5 decaf::net::NoRouteToHostException::NoRouteToHostException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.390.1.6 decaf::net::NoRouteToHostException::NoRouteToHostException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.390.1.7 virtual decaf::net::NoRouteToHostException::~NoRouteToHostException () throw () [inline, virtual]

6.390.2 Member Function Documentation

6.390.2.1 virtual NoRouteToHostException* decaf::net::NoRouteToHostException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2380).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**NoRouteToHostException.h**

6.391 decaf::security::NoSuchAlgorithmException Class Reference

`#include <src/main/decaf/security/NoSuchAlgorithmException.h>`Inheritance diagram for decaf::security::NoSuchAlgorithmException:

Public Member Functions

- **NoSuchAlgorithmException** () throw ()
  Default Constructor.
- **NoSuchAlgorithmException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **NoSuchAlgorithmException** (const **NoSuchAlgorithmException** &ex) throw ()
  Copy Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchAlgorithmException** (const std::exception *cause) throw ()
  Constructor.
- **NoSuchAlgorithmException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **NoSuchAlgorithmException** * **clone** () const
  Clones this exception.
- virtual ~**NoSuchAlgorithmException** () throw ()

6.391.1 Constructor & Destructor Documentation

6.391.1.1 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException () throw () [inline]

Default Constructor.

6.391.1.2 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.391.1.3 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const NoSuchAlgorithmException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.391.1.4 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.391.1.5 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.391.1.6 decaf::security::NoSuchAlgorithmException::NoSuchAlgorithmException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.391.1.7 virtual decaf::security::NoSuchAlgorithmException::~NoSuchAlgorithmException () throw () [inline, virtual]

6.391.2 Member Function Documentation

6.391.2.1 virtual NoSuchAlgorithmException* decaf::security::NoSuchAlgorithmException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1381).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchAlgorithmException.h**

6.392 decaf::lang::exceptions::NoSuchElementException Class Reference

`#include <src/main/decaf/lang/exceptions/NoSuchElementException.h>`Inheritance diagram for decaf::lang::exceptions::NoSuchElementException:

Public Member Functions

- **NoSuchElementException** () throw ()
  Default Constructor.
- **NoSuchElementException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **NoSuchElementException** (const **NoSuchElementException** &ex) throw ()
  Copy Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchElementException** (const std::exception ***cause**) throw ()
  Constructor.
- **NoSuchElementException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **NoSuchElementException** * **clone** () const
  Clones this exception.
- virtual ~**NoSuchElementException** () throw ()

6.392.1 Constructor & Destructor Documentation

6.392.1.1 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException () throw () `[inline]`

Default Constructor.

6.392.1.2 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.392.1.3 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const NoSuchElementException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.392.1.4 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.392.1.5 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.392.1.6 decaf::lang::exceptions::NoSuchElementException::NoSuchElementException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.392.1.7 virtual decaf::lang::exceptions::NoSuchElementException::~NoSuchElementException () throw () [inline, virtual]

6.392.2 Member Function Documentation

6.392.2.1 virtual NoSuchElementException* decaf::lang::exceptions::NoSuchElementException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NoSuchElementException.h**

6.393 decaf::security::NoSuchProviderException Class Reference

`#include <src/main/decaf/security/NoSuchProviderException.h>`Inheritance diagram for decaf::security::NoSuchProviderException:

Public Member Functions

- **NoSuchProviderException** () throw ()
  Default Constructor.
- **NoSuchProviderException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **NoSuchProviderException** (const **NoSuchProviderException** &ex) throw ()
  Copy Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **NoSuchProviderException** (const std::exception *cause) throw ()
  Constructor.
- **NoSuchProviderException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **NoSuchProviderException** * **clone** () const
  Clones this exception.
- virtual ~**NoSuchProviderException** () throw ()

6.393.1 Constructor & Destructor Documentation

6.393.1.1 decaf::security::NoSuchProviderException::NoSuchProviderException () throw () [inline]

Default Constructor.

6.393.1.2 decaf::security::NoSuchProviderException::NoSuchProviderException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.393.1.3 decaf::security::NoSuchProviderException::NoSuchProviderException (const NoSuchProviderException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.393.1.4 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.393.1.5 decaf::security::NoSuchProviderException::NoSuchProviderException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.393.1.6 decaf::security::NoSuchProviderException::NoSuchProviderException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.393.1.7 virtual decaf::security::NoSuchProviderException::~NoSuchProviderException () throw () [inline, virtual]

6.393.2 Member Function Documentation

6.393.2.1 virtual NoSuchProviderException* decaf::security::NoSuchProviderException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1381).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**NoSuchProviderException.h**

6.394 decaf::lang::exceptions::NullPointerException Class Reference

`#include <src/main/decaf/lang/exceptions/NullPointerException.h>`Inheritance diagram for decaf::lang::exceptions::NullPointerException:

Public Member Functions

- **NullPointerException** () throw ()
  Default Constructor.
- **NullPointerException** (const **Exception** &ex) throw ()
  *Conversion Constructor from some other **Exception** (p. 1268).*
- **NullPointerException** (const **NullPointerException** &ex) throw ()
  Copy Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **NullPointerException** (const std::exception ***cause**) throw ()
  Constructor.
- **NullPointerException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occured.
- virtual **NullPointerException** * **clone** () const
  Clones this exception.
- virtual ~**NullPointerException** () throw ()

6.394.1 Constructor & Destructor Documentation

6.394.1.1 decaf::lang::exceptions::NullPointerException::NullPointerException () throw () [inline]

Default Constructor.

6.394.1.2 decaf::lang::exceptions::NullPointerException::NullPointerException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.394.1.3 decaf::lang::exceptions::NullPointerException::NullPointerException (const NullPointerException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.394.1.4 decaf::lang::exceptions::NullPointerException::NullPointerException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.394.1.5 decaf::lang::exceptions::NullPointerException::NullPointerException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.394.1.6 decaf::lang::exceptions::NullPointerException::NullPointerException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.394.1.7 virtual decaf::lang::exceptions::NullPointerException::~NullPointerException () throw () [inline, virtual]

6.394.2 Member Function Documentation

6.394.2.1 virtual NullPointerException* decaf::lang::exceptions::NullPointerException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NullPointerException.h**

6.395 decaf::lang::Number Class Reference

The abstract class **Number** (p. 1954) is the superclass of classes **Byte** (p. 664), **Double** (p. 1231), **Float** (p. 1328), **Integer** (p. 1428), **Long** (p. 1652), and **Short** (p. 2321).

`#include <src/main/decaf/lang/Number.h>`Inheritance diagram for decaf::lang::Number:

Public Member Functions

- virtual ~**Number** ()
- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual double **doubleValue** () const =0

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const =0

  Answers the float value which the receiver represents.

- virtual int **intValue** () const =0

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const =0

  Answers the long value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

6.395.1 Detailed Description

The abstract class **Number** (p. 1954) is the superclass of classes **Byte** (p. 664), **Double** (p. 1231), **Float** (p. 1328), **Integer** (p. 1428), **Long** (p. 1652), and **Short** (p. 2321). Subclasses of **Number** (p. 1954) must provide methods to convert the represented numeric value to byte, double, float, int, long, and short.

6.395.2 Constructor & Destructor Documentation

6.395.2.1 virtual decaf::lang::Number::~Number () [inline, virtual]

6.395.3 Member Function Documentation

6.395.3.1 virtual unsigned char decaf::lang::Number::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

byte the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 666), **decaf::lang::Character** (p. 803), **decaf::lang::Double** (p. 1233), **decaf::lang::Float** (p. 1330), **decaf::lang::Integer** (p. 1431), **decaf::lang::Long** (p. 1655), and **decaf::lang::Short** (p. 2323).

6.395.3.2 virtual double decaf::lang::Number::doubleValue () const [pure virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 667), **decaf::lang::Character** (p. 804), **decaf::lang::Double** (p. 1235), **decaf::lang::Float** (p. 1331), **decaf::lang::Integer** (p. 1432), **decaf::lang::Long** (p. 1656), **decaf::lang::Short** (p. 2324), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 493).

6.395.3.3 virtual float decaf::lang::Number::floatValue () const [pure virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 668), **decaf::lang::Character** (p. 804), **decaf::lang::Double** (p. 1236), **decaf::lang::Float** (p. 1333), **decaf::lang::Integer** (p. 1433), **decaf::lang::Long** (p. 1657), **decaf::lang::Short** (p. 2325), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 493).

6.395.3.4 virtual int decaf::lang::Number::intValue () const [pure virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 668), **decaf::lang::Character** (p. 805), **decaf::lang::Double** (p. 1236), **decaf::lang::Float** (p. 1333), **decaf::lang::Integer** (p. 1433), **decaf::lang::Long** (p. 1657), **decaf::lang::Short** (p. 2325), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 495).

6.395.3.5 virtual long long decaf::lang::Number::longValue () const [pure virtual]

Answers the long value which the receiver represents.

Returns:

long long the value of the receiver.

Implemented in **decaf::lang::Byte** (p. 668), **decaf::lang::Character** (p. 806), **decaf::lang::Double** (p. 1237), **decaf::lang::Float** (p. 1334), **decaf::lang::Integer** (p. 1434), **decaf::lang::Long** (p. 1658), **decaf::lang::Short** (p. 2325), and **decaf::util::concurrent::atomic::AtomicInteger** (p. 495).

6.395.3.6 virtual short decaf::lang::Number::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

short the value of the receiver.

Reimplemented in **decaf::lang::Byte** (p. 670), **decaf::lang::Character** (p. 807), **decaf::lang::Double** (p. 1239), **decaf::lang::Float** (p. 1336), **decaf::lang::Integer** (p. 1438), **decaf::lang::Long** (p. 1662), and **decaf::lang::Short** (p. 2327).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Number.h**

6.396 decaf::lang::exceptions::NumberFormatException Class Reference

`#include <src/main/decaf/lang/exceptions/NumberFormatException.h>`Inheritance diagram for decaf::lang::exceptions::NumberFormatException:

Public Member Functions

- **NumberFormatException** ()

  Default Constructor.

- **NumberFormatException** (const **Exception** &ex) throw ()

  Conversion Constructor from some other ***Exception*** *(*p. *1268**).*

- **NumberFormatException** (const **NumberFormatException** &ex) throw ()

  Copy Constructor.

- **NumberFormatException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **NumberFormatException** (const std::exception *__cause__) throw ()

  Constructor.

- **NumberFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occured.

- virtual **NumberFormatException** * **clone** () const

  Clones this exception.

- virtual ~**NumberFormatException** () throw ()

6.396.1 Constructor & Destructor Documentation

6.396.1.1 decaf::lang::exceptions::NumberFormatException::NumberFormatException () `[inline]`

Default Constructor.

Referenced by clone().

6.396.1.2 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.396.1.3 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const NumberFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.396.1.4 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.396.1.5 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.396.1.6 decaf::lang::exceptions::NumberFormatException::NumberFormatException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occured. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

References decaf::lang::Exception::buildMessage(), and decaf::lang::Exception::setMark().

6.396.1.7 virtual decaf::lang::exceptions::NumberFormatException::~NumberFormatException () throw () [inline, virtual]

6.396.2 Member Function Documentation

6.396.2.1 virtual NumberFormatException* decaf::lang::exceptions::NumberFormatException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

References NumberFormatException().

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**NumberFormatException.h**

6.397 cms::ObjectMessage Class Reference

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

`#include <src/main/cms/ObjectMessage.h>`Inheritance diagram for cms::ObjectMessage:

Public Member Functions

- virtual ~**ObjectMessage** ()

6.397.1 Detailed Description

Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object. serialized `ObjectMessage` (p. 1960)s.

Since:

1.0

6.397.2 Constructor & Destructor Documentation

6.397.2.1 virtual cms::ObjectMessage::~ObjectMessage () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/cms/**ObjectMessage.h**

6.398 decaf::security_provider::unix::openssl::OpenSSLX500Principal Class Reference

The `OpenSSLX500Principal` (p. 1961) wraps around an OpenSSL `X509_NAME` structure.

`#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX500Principal.h>`

Public Member Functions

- **OpenSSLX500Principal** (X509_NAME *name)
  Constructor.
- virtual ~**OpenSSLX500Principal** ()
  Destructor.
- virtual X509_NAME * **getX509Name** ()
  Accessor to the underlying X509 name structure.
- virtual bool **equals** (const Principal &another) const
  Compares two principals to see if they are the same.
- virtual std::string **getName** () const
  Returns the distinguished name string using the RFC2253 formatting.
- virtual void **getEncoded** (std::vector< unsigned char > &output) const
  Serializes the distinguished name to its ASN.1 DER encoded form.

Static Public Member Functions

- static void **getEncoded** (X509_NAME *name, std::vector< unsigned char > &output)
  Serializes the given distinguished name to its ASN.1 DER encoded form.
- static std::string **toString** (X509_NAME *name) const
  Converts the given name to a string using the RFC2253 formatting.

6.398.1 Detailed Description

The `OpenSSLX500Principal` (p. 1961) wraps around an OpenSSL `X509_NAME` structure. It does not, however, control the lifetime of the structure.

6.398.2 Constructor & Destructor Documentation

6.398.2.1 decaf::security_provider::unix::openssl::OpenSSLX500Principal::OpenSSLX500Principal (X509_NAME * *name*)

Constructor. Saves the **internal** (p. 95) X509 name and caches the string representation of the name.

Parameters:

name The underlying X509 name structure.

6.398.2.2 virtual decaf::security_-provider::unix::openssl::OpenSSLX500Principal::~OpenSSLX500Principal () [inline, virtual]

Destructor. Does nothing.

6.398.3 Member Function Documentation

6.398.3.1 virtual bool decaf::security_-provider::unix::openssl::OpenSSLX500Principal::equals (const Principal & *another*) const [virtual]

Compares two principals to see if they are the same.

Parameters:

another A principal to be tested for equality to this one.

Returns:

true if the given principal is equivalent to this one.

6.398.3.2 static void decaf::security_-provider::unix::openssl::OpenSSLX500Principal::getEncoded (X509_NAME * *name*, std::vector< unsigned char > & *output*) [static]

Serializes the given distinguished name to its ASN.1 DER encoded form.

Parameters:

name the X509 name structure to be encoded.

output Receives the distinguished name in ASN.1 DER encoded form.

6.398.3.3 virtual void decaf::security_-provider::unix::openssl::OpenSSLX500Principal::getEncoded (std::vector< unsigned char > & *output*) const [inline, virtual]

Serializes the distinguished name to its ASN.1 DER encoded form.

Parameters:

output Receives the distinguished name in ASN.1 DER encoded form.

6.398.3.4 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX500Principal::getName () const [inline, virtual]

Returns the distinguished name string using the RFC2253 formatting.

Returns:

the RFC2253 formatted distinguished name string.

References toString().

6.398.3.5 virtual X509_NAME* decaf::security_provider::unix::openssl::OpenSSLX500Principal::getX509Name () [inline, virtual]

Accessor to the underlying X509 name structure.

6.398.3.6 static std::string decaf::security_provider::unix::openssl::OpenSSLX500Principal::toString (X509_NAME * *name*) const [static]

Converts the given name to a string using the RFC2253 formatting.

Parameters:

name the X509 name structure to be formatted.

Returns:

the RFC2253 formatted name string.

Referenced by getName().

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/**OpenSSLX500Principal.h**

6.399 decaf::security_provider::unix::openssl::OpenSSLX509Certificate Class Reference

`#include <src/main/decaf/security_provider/unix/openssl/OpenSSLX509Certificate.h>`Inheritance diagram for decaf::security_provider::unix::openssl::OpenSSLX509Certificate:

Public Member Functions

- virtual ~**OpenSSLX509Certificate** ()
- virtual bool **equals** (const Certificate &cert) const =0

  Compares the encoded form of the two certificates.

- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)

  Provides the encoded form of this certificate.

- virtual std::string **getType** () const =0

  Returns the type of this certificate.

- virtual PublicKey * **getPublicKey** ()=0

  Gets the public key of this certificate.

- virtual const PublicKey * **getPublicKey** () const =0

  Gets the public key of this certificate.

- virtual void **verify** (const PublicKey &publicKey) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)

  Verifies that this certificate was signed with the private key that corresponds to the specified public key.

- virtual void **verify** (const PublicKey &publicKey, const std::string &sigProvider) const =0 throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException)

  Verifies that this certificate was signed with the private key that corresponds to the specified public key.

- virtual std::string **toString** () const =0

  Returns a string representation of this certificate.

- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const **decaf::util::Date** &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0

- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
- virtual int **getVersion** () const =0

6.399.1 Constructor & Destructor Documentation

6.399.1.1 virtual decaf::security_provider::unix::openssl::OpenSSLX509Certificate::~OpenSSLX509Certificate () [virtual]

6.399.2 Member Function Documentation

6.399.2.1 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2728).

6.399.2.2 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::checkValidity () const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.3 virtual bool decaf::security_provider::unix::openssl::OpenSSLX509Certificate::equals (const Certificate & *cert*) const [pure virtual]

Compares the encoded form of the two certificates.

Parameters:

cert The certificate to be tested for equality with this certificate.

Returns:

true if the given certificate is equal to this certificate.

6.399.2.4 virtual int decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getBasicConstraints () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.5 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getEncoded (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Provides the encoded form of this certificate.

Parameters:

output Receives the encoded form of this certificate.

Exceptions:

CertificateEncodingException if an encoding error occurs

Implements **decaf::security::cert::Certificate** (p. 788).

6.399.2.6 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.7 virtual const X500Principal* decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getIssuerX500Principal () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.8 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getKeyUsage (std::vector< unsigned char > & *output*) const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.9 virtual Date decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getNotAfter () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.10 virtual Date decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getNotBefore () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.11 virtual const PublicKey* decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getPublicKey () const [pure virtual]

Gets the public key of this certificate.

Returns:

the public key

Implements **decaf::security::cert::Certificate** (p. 788).

6.399.2.12 virtual PublicKey* decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getPublicKey () [pure virtual]

Gets the public key of this certificate.

Returns:

the public key

Implements **decaf::security::cert::Certificate** (p. 788).

6.399.2.13 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgName () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2729).

6.399.2.14 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgOID () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.15 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSigAlgParams (std::vector< unsigned char > & *output*) const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.16 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSignature (std::vector< unsigned char > & *output*) const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.17 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSubjectUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.18 virtual const X500Principal* decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getSubjectX500Principal () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.19 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getTBSCertificate (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.20 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getType () const [pure virtual]

Returns the type of this certificate.

Returns:

the type of this certificate

Implements **decaf::security::cert::Certificate** (p. 789).

6.399.2.21 virtual int decaf::security_provider::unix::openssl::OpenSSLX509Certificate::getVersion () const [pure virtual]

Implements **decaf::security::cert::X509Certificate** (p. 2730).

6.399.2.22 virtual std::string decaf::security_provider::unix::openssl::OpenSSLX509Certificate::toString () const [pure virtual]

Returns a string representation of this certificate.

Returns:

a string representation of this certificate

Implements **decaf::security::cert::Certificate** (p. 789).

6.399.2.23 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::verify (const PublicKey & *publicKey*, const std::string & *sigProvider*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key. Uses the verification engine of the specified provider.

Parameters:

publicKey The public key used to carry out the validation.

sigProvider The name of the signature provider

Exceptions:

NoSuchAlgorithmException - on unsupported signature algorithms.

InvalidKeyException - on incorrect key.

NoSuchProviderException - if there's no default provider.

SignatureException - on signature errors.

CertificateException - on encoding errors.

6.399.2.24 virtual void decaf::security_provider::unix::openssl::OpenSSLX509Certificate::verify (const PublicKey & *publicKey*) const throw (NoSuchAlgorithmException, InvalidKeyException, NoSuchProviderException, SignatureException, CertificateException) [pure virtual]

Verifies that this certificate was signed with the private key that corresponds to the specified public key.

Parameters:

publicKey The public key used to carry out the validation.

Exceptions:

NoSuchAlgorithmException - on unsupported signature algorithms.

InvalidKeyException - on incorrect key.

NoSuchProviderException - if there's no default provider.

SignatureException - on signature errors.

CertificateException - on encoding errors.

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/unix/openssl/**OpenSSLX509Certificate.h**

6.400 activemq::wireformat::openwire::OpenWireFormat Class Reference

`#include <src/main/activemq/wireformat/openwire/OpenWireFormat.h>`Inheritance diagram for activemq::wireformat::openwire::OpenWireFormat:

Public Member Functions

- **OpenWireFormat** (const **decaf::util::Properties** &properties)

  *Constructs a new **OpenWireFormat** (p. 1970) object.*

- virtual ~**OpenWireFormat** ()
- virtual bool **hasNegotiator** () const

  *Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw (decaf::lang::exceptions::UnsupportedOperationException)

  If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

- void **addMarshaller** (**marshal::DataStreamMarshaller** *marshaler)

  Allows an external source to add marshalers to this object for types that may be marshaled or unmarhsaled.

- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (decaf::io::IOException)

  Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (decaf::io::IOException)

  Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form.

- virtual int **tightMarshalNestedObject1** (**commands::DataStructure** *object, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Utility method for Tight Marshaling the given object to the boolean stream passed.

- void **tightMarshalNestedObject2** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *ds, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  *Utility method that will Tight **marshal** (p. 76) some internally nested object that implements the DataStructure interface.*

- **commands::DataStructure** * **tightUnmarshalNestedObject** (**decaf::io::DataInputStream** *dis, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream.

- **commands::DataStructure** * **looseUnmarshalNestedObject** (**decaf::io::DataInputStream** *dis) throw (decaf::io::IOException)

  Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format.

- void **looseMarshalNestedObject** (**commands::DataStructure** *o, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Utility method to loosely Marshal an object that is derived from the DataStrucutre interface.

- void **renegotiateWireFormat** (const **commands::WireFormatInfo** &info) throw (decaf::lang::exceptions::IllegalStateException)

  Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

- virtual void **setPreferedWireFormatInfo** (const **Pointer< commands::WireFormatInfo >** &info) throw (decaf::lang::exceptions::IllegalStateException)

  Configures this object using the provided WireformatInfo object.

- virtual const **Pointer< commands::WireFormatInfo >** & **getPreferedWireFormatInfo** () const

  Gets the Preferred WireFormatInfo object that this class holds.

- bool **isStackTraceEnabled** () const

  Checks if the stackTraceEnabled flag is on.

- void **setStackTraceEnabled** (bool stackTraceEnabled)

  Sets if the stackTraceEnabled flag is on.

- bool **isTcpNoDelayEnabled** () const

  Checks if the tcpNoDelayEnabled flag is on.

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

  Sets if the tcpNoDelayEnabled flag is on.

- int **getVersion** () const

  Get the current Wireformat Version.

- void **setVersion** (int version) throw (decaf::lang::exceptions::IllegalArgumentException)

  Set the current Wireformat Version.

- bool **isCacheEnabled** () const

  Checks if the cacheEnabled flag is on.

- void **setCacheEnabled** (bool cacheEnabled)

Sets if the cacheEnabled flag is on.

- int **getCacheSize** () const

  Returns the currently set Cache size.

- void **setCacheSize** (int value)

  Sets the current Cache size.

- bool **isTightEncodingEnabled** () const

  Checks if the tightEncodingEnabled flag is on.

- void **setTightEncodingEnabled** (bool tightEncodingEnabled)

  Sets if the tightEncodingEnabled flag is on.

- bool **isSizePrefixDisabled** () const

  Checks if the sizePrefixDisabled flag is on.

- void **setSizePrefixDisabled** (bool sizePrefixDisabled)

  Sets if the sizePrefixDisabled flag is on.

- long long **getMaxInactivityDuration** () const

  Gets the MaxInactivityDuration setting.

- void **setMaxInactivityDuration** (long long value)

  Sets the MaxInactivityDuration setting.

- long long **getMaxInactivityDurationInitialDelay** () const

  Gets the MaxInactivityDurationInitialDelay setting.

- void **setMaxInactivityDurationInitialDelay** (long long value)

  Sets the MaxInactivityDurationInitialDelay setting.

Protected Member Functions

- **commands::DataStructure** * **doUnmarshal** (**decaf::io::DataInputStream** *dis) throw (decaf::io::IOException)

  Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object.

- void **destroyMarshalers** ()

  Cleans up all registered Marshallers and empties the dataMarshallers vector.

Static Protected Attributes

- static const unsigned char **NULL_TYPE**
- static const int **DEFAULT_VERSION** = 1

6.400.1 Constructor & Destructor Documentation

6.400.1.1 activemq::wireformat::openwire::OpenWireFormat::OpenWireFormat (const decaf::util::Properties & *properties*)

Constructs a new **OpenWireFormat** (p. 1970) object.

Parameters:

properties - can contain optional config params.

6.400.1.2 virtual activemq::wireformat::openwire::OpenWireFormat::~OpenWireFormat () [virtual]

6.400.2 Member Function Documentation

6.400.2.1 void activemq::wireformat::openwire::OpenWireFormat::addMarshaller (marshal::DataStreamMarshaller * *marshaler*)

Allows an external source to add marshalers to this object for types that may be marshaled or unmarhsaled.

Parameters:

marshaler - the Marshaler to add to the collection.

6.400.2.2 virtual Pointer<transport::Transport> activemq::wireformat::openwire::OpenWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns:

new instance of a **WireFormatNegotiator** (p. 2721).

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.400.2.3 void activemq::wireformat::openwire::OpenWireFormat::destroyMarshalers () [protected]

Cleans up all registered Marshallers and empties the dataMarshallers vector. This should be called before a reconfiguration of the version marshallers, or on destruction of this object

6.400.2.4 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::doUnmarshal (decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException) [protected]

Perform the actual unmarshal of data from the given DataInputStream return the unmarshalled DataStrucutre object once done, caller takes ownership of this object. This method can return null if the type of the object to unmarshal is NULL, empty data.

Parameters:

dis - DataInputStream to read from

Returns:

new DataStructure* that the caller owns

Exceptions:

IOException if an error occurs during the unmarshal

6.400.2.5 int activemq::wireformat::openwire::OpenWireFormat::getCacheSize () const [inline]

Returns the currently set Cache size.

Returns:

the current value of the broker's cache size.

6.400.2.6 long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDuration () const [inline]

Gets the MaxInactivityDuration setting.

Returns:

maximum inactivity duration value in milliseconds.

6.400.2.7 long long activemq::wireformat::openwire::OpenWireFormat::getMaxInactivityDurationInitialDelay () const [inline]

Gets the MaxInactivityDurationInitialDelay setting.

Returns:

maximum inactivity duration initial delay value in milliseconds.

6.400.2.8 virtual const Pointer<commands::WireFormatInfo>& activemq::wireformat::openwire::OpenWireFormat::getPreferedWireFormatInfo () const [inline, virtual]

Gets the Preferred WireFormatInfo object that this class holds.

Returns:

pointer to a preferred WireFormatInfo object

6.400.2.9 int activemq::wireformat::openwire::OpenWireFormat::getVersion () const [inline, virtual]

Get the current Wireformat Version.

Returns:

int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.400.2.10 virtual bool activemq::wireformat::openwire::OpenWireFormat::hasNegotiator () const [inline, virtual]

Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 2693) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.400.2.11 bool activemq::wireformat::openwire::OpenWireFormat::isCacheEnabled () const [inline]

Checks if the cacheEnabled flag is on.

Returns:

true if the flag is on.

6.400.2.12 bool activemq::wireformat::openwire::OpenWireFormat::isSizePrefixDisabled () const [inline]

Checks if the sizePrefixDisabled flag is on.

Returns:

true if the flag is on.

6.400.2.13 bool activemq::wireformat::openwire::OpenWireFormat::isStackTraceEnabled () const [inline]

Checks if the stackTraceEnabled flag is on.

Returns:

true if the flag is on.

6.400.2.14 bool activemq::wireformat::openwire::OpenWireFormat::isTcpNoDelayEnabled () const [inline]

Checks if the tcpNoDelayEnabled flag is on.

Returns:

true if the flag is on.

6.400.2.15 bool activemq::wireformat::openwire::OpenWireFormat::isTightEncodingEnabled () const [inline]

Checks if the tightEncodingEnabled flag is on.

Returns:

true if the flag is on.

6.400.2.16 void activemq::wireformat::openwire::OpenWireFormat::looseMarshalNestedObject (commands::DataStructure * *o*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException)

Utility method to loosely Marshal an object that is derived from the DataStrucutre interface. The marshaled data is written to the passed in DataOutputStream.

Parameters:

o - DataStructure derived Object to Marshal

dataOut - DataOutputStream to write the data to

Exceptions:

IOException if an error occurs.

6.400.2.17 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::looseUnmarshalNestedObject (decaf::io::DataInputStream * *dis*) throw (decaf::io::IOException)

Utility method to unmarshal an DataStructure object from an DataInputStream using the Loose Unmarshaling format. Will read the Data and construct a new DataStructure based Object, the pointer to the Object returned is now owned by the caller.

Parameters:

dis - the DataInputStream to read the data from

Returns:

a new DataStructure derived Object pointer

Exceptions:

IOException if an error occurs.

6.400.2.18 virtual void activemq::wireformat::openwire::OpenWireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) throw (decaf::io::IOException) [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal.

transport *(*p. *67)* The Transport instance that called this method.

out The output stream to write the command to.

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 2695).

6.400.2.19 void activemq::wireformat::openwire::OpenWireFormat::renegotiateWireFormat (const commands::WireFormatInfo & *info*) throw (decaf::lang::exceptions::IllegalStateException)

Called to re-negotiate the settings for the WireFormatInfo, these determine how the client and broker communicate.

Parameters:

info - The new Wireformat Info settings

Exceptions:

IllegalStateException is wire format can't be negotiated.

6.400.2.20 void activemq::wireformat::openwire::OpenWireFormat::setCacheEnabled (bool *cacheEnabled*) [inline]

Sets if the cacheEnabled flag is on.

Parameters:

cacheEnabled - true to turn flag is on

6.400.2.21 void activemq::wireformat::openwire::OpenWireFormat::setCacheSize (int *value*) [inline]

Sets the current Cache size.

Parameters:

value - the value to send as the broker's cache size.

6.400.2.22 void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDuration (long long *value*) [inline]

Sets the MaxInactivityDuration setting.

Parameters:

value - the Max inactivity duration value in milliseconds.

6.400.2.23 void activemq::wireformat::openwire::OpenWireFormat::setMaxInactivityDurationInitialDelay (long long *value*) [inline]

Sets the MaxInactivityDurationInitialDelay setting.

Parameters:

value - the Max inactivity Initial Delay duration value in milliseconds.

6.400.2.24 virtual void activemq::wireformat::openwire::OpenWireFormat::setPreferedWireFormatInfo (const Pointer< commands::WireFormatInfo > & *info*) throw (decaf::lang::exceptions::IllegalStateException) [virtual]

Configures this object using the provided WireformatInfo object.

Parameters:

info - a WireFormatInfo object, takes ownership.

6.400.2.25 void activemq::wireformat::openwire::OpenWireFormat::setSizePrefixDisabled (bool *sizePrefixDisabled*) [inline]

Sets if the sizePrefixDisabled flag is on.

Parameters:

sizePrefixDisabled - true to turn flag is on

6.400.2.26 void activemq::wireformat::openwire::OpenWireFormat::setStackTraceEnabled (bool *stackTraceEnabled*) [inline]

Sets if the stackTraceEnabled flag is on.

Parameters:

stackTraceEnabled - true to turn flag is on

6.400.2.27 void activemq::wireformat::openwire::OpenWireFormat::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*) [inline]

Sets if the tcpNoDelayEnabled flag is on.

Parameters:

tcpNoDelayEnabled - true to turn flag is on

6.400.2.28 void activemq::wireformat::openwire::OpenWireFormat::setTightEncodingEnabled (bool *tightEncodingEnabled*) [inline]

Sets if the tightEncodingEnabled flag is on.

Parameters:

tightEncodingEnabled - true to turn flag is on

6.400.2.29 void activemq::wireformat::openwire::OpenWireFormat::setVersion (int *version*) throw (decaf::lang::exceptions::IllegalArgumentException) [virtual]

Set the current Wireformat Version.

Parameters:

version - int that identifies the version

Implements **activemq::wireformat::WireFormat** (p. 2695).

6.400.2.30 virtual int activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject1 (commands::DataStructure * *object*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Utility method for Tight Marshaling the given object to the boolean stream passed.

Parameters:

object - The DataStructure to **marshal** (p. 76)

bs - the BooleanStream to write to

Returns:

size of the data returned.

6.400.2.31 void activemq::wireformat::openwire::OpenWireFormat::tightMarshalNestedObject2 (commands::DataStructure * *o*, decaf::io::DataOutputStream * *ds*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)

Utility method that will Tight **marshal** (p. 76) some internally nested object that implements the DataStructure interface. Writes the data to the Data Output Stream provided.

Parameters:

o - DataStructure object

ds - DataOuputStream for writing

bs - BooleanStream

Exceptions:

IOException if an error occurs.

6.400.2.32 commands::DataStructure* activemq::wireformat::openwire::OpenWireFormat::tightUnmarshalNestedObject (decaf::io::DataInputStream * *dis*, utils::BooleanStream * *bs*) throw (decaf::io::IOException)

Utility method used to Unmarshal a Nested DataStructure type object from the given DataInputStream. The DataStructure instance that is returned is now the property of the caller.

Parameters:

dis - DataInputStream to read from

bs - BooleanStream to read from

Returns:

Newly allocated DataStructure Object

Exceptions:

IOException if an error occurs.

6.400.2.33 virtual Pointer<commands::Command> activemq::wireformat::openwire::OpenWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and un-marshaled into the correct form. Returns a Pointer to the newly un-marshaled Command.

Parameters:

transport *(p. 67)* - Pointer to the **transport** (p. 67) that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 2695).

6.400.3 Field Documentation

6.400.3.1 const int activemq::wireformat::openwire::OpenWireFormat::DEFAULT_VERSION = 1 [static, protected]

6.400.3.2 const unsigned char activemq::wireformat::openwire::OpenWireFormat::NULL_TYPE [static, protected]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormat.h**

6.401 activemq::wireformat::openwire::OpenWireFormatFactory Class Reference

`#include <src/main/activemq/wireformat/openwire/OpenWireFormatFactory.h>`Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatFactory:

Public Member Functions

- **OpenWireFormatFactory** ()

  Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called.

- virtual ~**OpenWireFormatFactory** ()
- virtual **Pointer< wireformat::WireFormat > createWireFormat** (const **decaf::util::Properties** &properties) throw (decaf::lang::exceptions::IllegalStateException)

  *Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.*

6.401.1 Constructor & Destructor Documentation

6.401.1.1 activemq::wireformat::openwire::OpenWireFormatFactory::OpenWireFormatFactory () `[inline]`

Constructor - Sets Defaults for all properties, these are all subject to change once the `createWireFormat` method is called. URL options ---------- wireFormat.stackTraceEnabled wireFormat.cacheEnabled wireFormat.tcpNoDelayEnabled wireFormat.tightEncodingEnabled wireFormat.sizePrefixDisabled wireFormat.maxInactivityDuration wireFormat.maxInactivityDurationInitialDelay

6.401.1.2 virtual activemq::wireformat::openwire::OpenWireFormatFactory::~OpenWireFormatFactory () `[inline, virtual]`

6.401.2 Member Function Documentation

6.401.2.1 virtual Pointer<wireformat::WireFormat> activemq::wireformat::openwire::OpenWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalStateException) `[virtual]`

Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the Properties for this **WireFormat** (p. 2693)

Implements **activemq::wireformat::WireFormatFactory** (p. 2697).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatFactory.h**

6.402 activemq::wireformat::openwire::OpenWireFormatNegotiator Class Reference

`#include <src/main/activemq/wireformat/openwire/OpenWireFormatNegotiator.h>`Inheritance diagram for activemq::wireformat::openwire::OpenWireFormatNegotiator:

Public Member Functions

- **OpenWireFormatNegotiator** (**OpenWireFormat** *wireFormat, const **Pointer< transport::Transport >** &**next**)

  Constructor - Initializes this object around another Transport.

- virtual ~**OpenWireFormatNegotiator** ()
- virtual void **oneway** (const **Pointer< commands::Command >** &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer< commands::Response > request** (const **Pointer< commands::Command >** &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given request to the server and waits for the response.

- virtual **Pointer< commands::Response > request** (const **Pointer< commands::Command >** &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given request to the server and waits for the response.

- virtual void **onCommand** (const **Pointer< commands::Command >** &command)

  This is called in the context of the nested transport's reading thread.

- virtual void **onTransportException** (**transport::Transport** *source, const **decaf::lang::Exception** &ex)

  *Event handler for an exception from a command **transport*** (p. 67).

- virtual void **start** () throw (cms::CMSException)

  *Starts this **transport*** (p. 67) *object and creates the thread for polling on the input stream for **commands*** (p. 59).

- virtual void **close** () throw (cms::CMSException)

  Stops the polling thread and closes the streams.

6.402.1 Constructor & Destructor Documentation

6.402.1.1 activemq::wireformat::openwire::OpenWireFormatNegotiator::OpenWireFormatNegotiator (OpenWireFormat * *wireFormat*, const Pointer< transport::Transport > & *next*)

Constructor - Initializes this object around another Transport.

Parameters:

wireFormat - The **WireFormat** (p. 2693) object we use to negotiate

next - The next **transport** (p. 67) in the chain

6.402.1.2 virtual activemq::wireformat::openwire::OpenWireFormatNegotiator::~OpenWireFormatNegotiat () [virtual]

6.402.2 Member Function Documentation

6.402.2.1 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::close () throw (cms::CMSException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

CMSException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 2618).

6.402.2.2 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onCommand (const Pointer< commands::Command > & *command*) [virtual]

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters:

command the received from the nested **transport** (p. 67).

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.402.2.3 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::oneway (const Pointer< commands::Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.402.2.4 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::onTransportException (transport::Transport * *source*, const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

source The source of the exception

ex The exception.

6.402.2.5 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command The request to send.

timeout The time to wait for the response.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 2621).

6.402.2.6 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireFormatNegotiator::request (const Pointer< commands::Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response. First waits for the WireFormatInfo exchange to happen so that we know how to encode out-bound data.

Parameters:

command The request to send.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 2622).

6.402.2.7 virtual void activemq::wireformat::openwire::OpenWireFormatNegotiator::start () throw (cms::CMSException) [virtual]

Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this **transport** (p. 67) has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 2622).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireFormatNegotiator.h**

6.403 activemq::wireformat::openwire::OpenWireResponseBuilder Class Reference

`#include <src/main/activemq/wireformat/openwire/OpenWireResponseBuilder.h>`Inheritance diagram for activemq::wireformat::openwire::OpenWireResponseBuilder:

Public Member Functions

- **OpenWireResponseBuilder** ()
- virtual ~**OpenWireResponseBuilder** ()
- virtual **Pointer< commands::Response > buildResponse** (const **Pointer< commands::Command >** &command)

  Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const **Pointer< commands::Command >** &command, **decaf::util::StlQueue< Pointer< commands::Command > >** &queue)

  *When called the ResponseBuilder must construct all the Responses or Asynchronous **commands** (p. 59) that would be sent to this client by the Broker upon receipt of the passed command.*

6.403.1 Constructor & Destructor Documentation

6.403.1.1 activemq::wireformat::openwire::OpenWireResponseBuilder::OpenWireResponseBuilder () [inline]

6.403.1.2 virtual activemq::wireformat::openwire::OpenWireResponseBuilder::~OpenWireResponseBuilder () [inline, virtual]

6.403.2 Member Function Documentation

6.403.2.1 virtual void activemq::wireformat::openwire::OpenWireResponseBuilder::buildIncomingCommands (const Pointer< commands::Command > & *command*, decaf::util::StlQueue< Pointer< commands::Command > > & *queue*) [virtual]

When called the ResponseBuilder must construct all the Responses or Asynchronous **commands** (p. 59) that would be sent to this client by the Broker upon receipt of the passed command.

Parameters:

command - The Command being sent to the Broker.

queue - Queue of Command sent back from the broker.

Implements **activemq::transport::mock::ResponseBuilder** (p. 2235).

6.403.2.2 virtual Pointer<commands::Response> activemq::wireformat::openwire::OpenWireResponseBuilder::buildResponse (const Pointer< commands::Command > & *command*) [virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters:

command - The command to build a response for

Returns:

A Response object pointer, or NULL if no response.

Implements **activemq::transport::mock::ResponseBuilder** (p. 2236).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/**OpenWireResponseBuilder.h**

6.404 activemq::wireformat::openwire::utils::OpenwireStringSupport Class Reference

`#include <src/main/activemq/wireformat/openwire/utils/OpenwireStringSupport.h>`

Static Public Member Functions

- static std::string **readString** (**decaf::io::DataInputStream** &dataIn) throw (decaf::io::IOException)

  Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

- static void **writeString** (**decaf::io::DataOutputStream** &dataOut, const std::string *str) throw (decaf::io::IOException)

  Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Protected Member Functions

- **OpenwireStringSupport** ()
- virtual ~**OpenwireStringSupport** ()

6.404.1 Constructor & Destructor Documentation

6.404.1.1 activemq::wireformat::openwire::utils::OpenwireStringSupport::OpenwireStringSupport () [inline, protected]

6.404.1.2 virtual activemq::wireformat::openwire::utils::OpenwireStringSupport::~OpenwireStringSupport () [inline, protected, virtual]

6.404.2 Member Function Documentation

6.404.2.1 static std::string activemq::wireformat::openwire::utils::OpenwireStringSupport::readString (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static]

Static method used for reading a string that uses the Openwire format from a DataInputStream, this can throw an IOException for the same reason as a DataInputStream.readUTF might, as well as if the string that is received doesn't conform to the supported character encoding.

Parameters:

dataIn - DataInputStream to read from

Returns:

A string that has been read

Exceptions:

IOException on Error.

6.404.2.2 static void activemq::wireformat::openwire::utils::OpenwireStringSupport::writeString (decaf::io::DataOutputStream & *dataOut*, const std::string * *str*) throw (decaf::io::IOException) [static]

Static method used for writing a string that uses the Openwire format from a DataOutputStream, this can throw an IOException for the same reason as a DataOutputStream.writeUTF might.

Parameters:

dataOut - DataOutputStream to write to

str - A pointer to a string that should be written, NULL needs to be an option here as its written as -1.

Exceptions:

IOException on Error.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/utils/**OpenwireStringSupport.h**

6.405 decaf::io::OutputStream Class Reference

Base interface for an output stream.

`#include <src/main/decaf/io/OutputStream.h>`Inheritance diagram for decaf::io::OutputStream:

Public Member Functions

- virtual ~**OutputStream** ()
- virtual void **write** (unsigned char c)=0 throw (IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer)=0 throw (IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len)=0 throw (IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** ()=0 throw (IOException)

  Flushes any pending writes in this output stream.

6.405.1 Detailed Description

Base interface for an output stream.

6.405.2 Constructor & Destructor Documentation

6.405.2.1 virtual decaf::io::OutputStream::~OutputStream () `[inline, virtual]`

6.405.3 Member Function Documentation

6.405.3.1 virtual void decaf::io::OutputStream::flush () throw (IOException) `[pure virtual]`

Flushes any pending writes in this output stream.

Exceptions:

IOException *(*p. *1477)*

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 2400), **decaf::internal::io::StandardOutputStream** (p. 2410), **decaf::io::BufferedOutputStream** (p. 639), **decaf::io::ByteArrayOutputStream** (p. 725), **decaf::io::FilterOutputStream** (p. 1324), and **decaf::net::SocketOutputStream** (p. 2391).

6.405.3.2 virtual void decaf::io::OutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implemented in **activemq::io::LoggingOutputStream** (p. 1635), **decaf::internal::io::StandardErrorOutputStream** (p. 2401), **decaf::internal::io::StandardOutputStream** (p. 2411), **decaf::io::BufferedOutputStream** (p. 639), **decaf::io::ByteArrayOutputStream** (p. 727), **decaf::io::DataOutputStream** (p. 1127), **decaf::io::FilterOutputStream** (p. 1326), and **decaf::net::SocketOutputStream** (p. 2393).

6.405.3.3 virtual void decaf::io::OutputStream::write (const std::vector< unsigned char > & *buffer*) throw (IOException) [pure virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implemented in **decaf::internal::io::StandardErrorOutputStream** (p. 2402), **decaf::internal::io::StandardOutputStream** (p. 2412), **decaf::io::BufferedOutputStream** (p. 640), **decaf::io::ByteArrayOutputStream** (p. 727), **decaf::io::DataOutputStream** (p. 1127), **decaf::io::FilterOutputStream** (p. 1326), and **decaf::net::SocketOutputStream** (p. 2393).

6.405.3.4 virtual void decaf::io::OutputStream::write (unsigned char *c*) throw (IOException) [pure virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException *(p. 1477)* thrown if an error occurs.

Implemented in **activemq::io::LoggingOutputStream** (p. 1636), **decaf::internal::io::StandardErrorOutputStream** (p. 2402), **decaf::internal::io::StandardOutputStream** (p. 2412), **decaf::io::BufferedOutputStream** (p. 640), **decaf::io::ByteArrayOutputStream** (p. 728), **decaf::io::DataOutputStream** (p. 1127), **decaf::io::FilterOutputStream** (p. 1327), and **decaf::net::SocketOutputStream** (p. 2393).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**OutputStream.h**

6.406 activemq::commands::PartialCommand Class Reference

`#include <src/main/activemq/commands/PartialCommand.h>`Inheritance diagram for activemq::commands::PartialCommand:

Public Member Functions

- **PartialCommand** ()
- virtual ~**PartialCommand** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **PartialCommand** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual int **getCommandId** () const
- virtual void **setCommandId** (int **commandId**)
- virtual const std::vector< unsigned char > & **getData** () const
- virtual std::vector< unsigned char > & **getData** ()
- virtual void **setData** (const std::vector< unsigned char > &**data**)

Static Public Attributes

- static const unsigned char **ID_PARTIALCOMMAND** = 60

Protected Member Functions

- **PartialCommand** (const **PartialCommand** &)
- **PartialCommand** & **operator**= (const **PartialCommand** &)

Protected Attributes

- int **commandId**
- std::vector< unsigned char > **data**

6.406.1 Constructor & Destructor Documentation

6.406.1.1 activemq::commands::PartialCommand::PartialCommand (const PartialCommand &) [inline, protected]

6.406.1.2 activemq::commands::PartialCommand::PartialCommand ()

6.406.1.3 virtual activemq::commands::PartialCommand::~PartialCommand () [virtual]

6.406.2 Member Function Documentation

6.406.2.1 virtual PartialCommand* activemq::commands::PartialCommand::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1576).

6.406.2.2 virtual void activemq::commands::PartialCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1576).

6.406.2.3 virtual bool activemq::commands::PartialCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1576).

6.406.2.4 virtual int activemq::commands::PartialCommand::getCommandId () const [virtual]

6.406.2.5 virtual std::vector<unsigned char>& activemq::commands::PartialCommand::getData () [virtual]

6.406.2.6 virtual const std::vector<unsigned char>& activemq::commands::PartialCommand::getData () const [virtual]

6.406.2.7 virtual unsigned char activemq::commands::PartialCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1577).

6.406.2.8 PartialCommand& activemq::commands::PartialCommand::operator= (const PartialCommand &) [inline, protected]

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1577).

6.406.2.9 virtual void activemq::commands::PartialCommand::setCommandId (int *commandId*) [virtual]

6.406.2.10 virtual void activemq::commands::PartialCommand::setData (const std::vector< unsigned char > & *data*) [virtual]

6.406.2.11 virtual std::string activemq::commands::PartialCommand::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

Reimplemented in **activemq::commands::LastPartialCommand** (p. 1577).

6.406.3 Field Documentation

6.406.3.1 int activemq::commands::PartialCommand::commandId [protected]

6.406.3.2 std::vector<unsigned char> activemq::commands::PartialCommand::data [protected]

6.406.3.3 const unsigned char activemq::commands::PartialCommand::ID_PARTIALCOMMAND = 60 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**PartialCommand.h**

6.407 activemq::wireformat::openwire::marshal::v2::PartialCommandMa Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1999).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual ~**PartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.407.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 1999). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.407.2 Constructor & Destructor Documentation

6.407.2.1 activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::PartialCommar () [inline]

6.407.2.2 virtual activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::~PartialComma () [inline, virtual]

6.407.3 Member Function Documentation

6.407.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1587).

6.407.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1587).

6.407.3.3 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1587).

6.407.3.4 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1588).

6.407.3.5 virtual int activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1588).

6.407.3.6 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1589).

6.407.3.7 virtual void activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller** (p. 1589).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**PartialCommandMarshaller.h**

6.408 activemq::wireformat::openwire::marshal::v3::PartialCommandMa Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2003).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual ~**PartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.408.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2003). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.408.2 Constructor & Destructor Documentation

6.408.2.1 activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::PartialCommar () [inline]

6.408.2.2 virtual activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::~PartialComma () [inline, virtual]

6.408.3 Member Function Documentation

6.408.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579).

6.408.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579).

6.408.3.3 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1579).

6.408.3.4 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1580).

6.408.3.5 virtual int activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1580).

6.408.3.6 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1581).

6.408.3.7 virtual void activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller** (p. 1581).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**PartialCommandMarshaller.h**

6.409 activemq::wireformat::openwire::marshal::v1::PartialCommandMa Class Reference

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2007).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller:

Public Member Functions

- **PartialCommandMarshaller** ()
- virtual ~**PartialCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.409.1 Detailed Description

Marshaling code for Open Wire Format for **PartialCommandMarshaller** (p. 2007). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.409.2 Constructor & Destructor Documentation

6.409.2.1 activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::PartialCommar () [inline]

6.409.2.2 virtual activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::~PartialComma () [inline, virtual]

6.409.3 Member Function Documentation

6.409.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1583).

6.409.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1583).

6.409.3.3 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1583).

6.409.3.4 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1584).

6.409.3.5 virtual int activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1584).

6.409.3.6 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1585).

6.409.3.7 virtual void activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller** (p. 1585).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**PartialCommandMarshaller.h**

6.410 decaf::lang::Pointer< T, REFCOUNTER > Class Template Reference

Decaf's implementation of a Smart **Pointer** (p. 2011) that is a template on a Type and is **Thread** (p. 2544) Safe if the default Reference Counter is used.

`#include <src/main/decaf/lang/Pointer.h>`

Public Types

- typedef T * **PointerType**
- typedef T & **ReferenceType**
- typedef REFCOUNTER **CounterType**

Public Member Functions

- **Pointer** ()

  Default Constructor.

- **Pointer** (const **PointerType** value)

  *Explicit Constructor, creates a **Pointer** (p. 2011) that contains value with a single reference.*

- **Pointer** (const **Pointer** &value) throw ()

  Copy constructor.

- template<typename T1 , typename R1 >
  Pointer (const **Pointer**< T1, R1 > &value) throw ()

  Copy constructor.

- template<typename T1 , typename R1 >
  Pointer (const **Pointer**< T1, R1 > &value, const **STATIC_CAST_TOKEN** &) throw ()

  Static Cast constructor.

- template<typename T1 , typename R1 >
  Pointer (const **Pointer**< T1, R1 > &value, const **DYNAMIC_CAST_TOKEN** &) throw (decaf::lang::exceptions::ClassCastException)

  Dynamic Cast constructor.

- virtual ~**Pointer** () throw ()
- void **reset** (T *value)

  *Resets the **Pointer** (p. 2011) to hold the new value.*

- **PointerType get** () const

  *Gets the real pointer that is contained within this **Pointer** (p. 2011).*

- void **swap** (**Pointer** &value) throw ()

  ***Exception** (p. 1268) Safe Swap Function.*

- **Pointer** & **operator**= (const **Pointer** &right) throw ()

Assigns the value of right to this ***Pointer*** *(p. 2011) and increments the reference Count.*

- template<typename T1 , typename R1 >
  Pointer & **operator**= (const **Pointer**< T1, R1 > &right) throw ()
- **ReferenceType operator*** ()

  Dereference Operator, returns a reference to the Contained value.

- **ReferenceType operator*** () const
- **PointerType operator->** ()

  Indirection Operator, returns a pointer to the Contained value.

- **PointerType operator->** () const
- bool **operator!** () const
- template<typename T1 , typename R1 >
  bool **operator==** (const **Pointer**< T1, R1 > &right) const
- template<typename T1 , typename R1 >
  bool **operator!=** (const **Pointer**< T1, R1 > &right) const
- template<typename T1 >
  Pointer< T1, **CounterType** > **dynamicCast** () const
- template<typename T1 >
  Pointer< T1, **CounterType** > **staticCast** () const

Friends

- bool **operator==** (const **Pointer** &left, const T *right)
- bool **operator==** (const T *left, const **Pointer** &right)
- bool **operator!=** (const **Pointer** &left, const T *right)
- bool **operator!=** (const T *left, const **Pointer** &right)

6.410.1 Detailed Description

template<typename T, typename REFCOUNTER = AtomicRefCounter> class decaf::lang::Pointer< T, REFCOUNTER >

Decaf's implementation of a Smart **Pointer** (p. 2011) that is a template on a Type and is **Thread** (p. 2544) Safe if the default Reference Counter is used. This **Pointer** (p. 2011) type allows for the substitution of different Reference Counter implementations which provide a means of using invasive reference counting if desired using a custom implementation of `ReferenceCounter`.

The Decaf smart pointer provide comparison operators for comparing **Pointer** (p. 2011) instances in the same manner as normal pointer, except that it does not provide an overload of operators (<, <=, >, >=). To allow use of a **Pointer** (p. 2011) in a STL container that requires it, **Pointer** (p. 2011) provides an implementation of std::less.

Since:

1.0

6.410.2 Member Typedef Documentation

6.410.2.1 template<typename T, typename REFCOUNTER = AtomicRefCounter> typedef REFCOUNTER decaf::lang::Pointer< T, REFCOUNTER >::CounterType

6.410.2.2 template<typename T, typename REFCOUNTER = AtomicRefCounter> typedef T* decaf::lang::Pointer< T, REFCOUNTER >::PointerType

6.410.2.3 template<typename T, typename REFCOUNTER = AtomicRefCounter> typedef T& decaf::lang::Pointer< T, REFCOUNTER >::ReferenceType

6.410.3 Constructor & Destructor Documentation

6.410.3.1 template<typename T, typename REFCOUNTER = AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer () [inline]

Default Constructor. Initialized the contained pointer to NULL, using the -> operator results in an exception unless reset to contain a real value.

Referenced by decaf::lang::Pointer< TransactionId >::reset().

6.410.3.2 template<typename T, typename REFCOUNTER = AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const PointerType *value*) [inline, explicit]

Explicit Constructor, creates a **Pointer** (p. 2011) that contains value with a single reference. This object now has ownership until a call to release.

Parameters:

value - instance of the type we are containing here.

6.410.3.3 template<typename T, typename REFCOUNTER = AtomicRefCounter> decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T, REFCOUNTER > & *value*) throw () [inline]

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

6.410.3.4 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & *value*) throw () [inline]

Copy constructor. Copies the value contained in the pointer to the new instance and increments the reference counter.

6.410.3.5 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & *value*, const STATIC_CAST_TOKEN &) throw () [inline]

Static Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a static cast on the value contained in the source **Pointer** (p. 2011) object.

Parameters:

value - **Pointer** (p. 2011) instance to cast to this type.

6.410.3.6 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > decaf::lang::Pointer< T, REFCOUNTER >::Pointer (const Pointer< T1, R1 > & *value*, const DYNAMIC_CAST_TOKEN &) throw (decaf::lang::exceptions::ClassCastException) [inline]

Dynamic Cast constructor. Copies the value contained in the pointer to the new instance and increments the reference counter performing a dynamic cast on the value contained in the source **Pointer** (p. 2011) object. If the cast fails and return NULL then this method throws a ClassCastException.

Parameters:

value - **Pointer** (p. 2011) instance to cast to this type.

Exceptions:

ClassCastException if the dynamic cast returns NULL

6.410.3.7 template<typename T, typename REFCOUNTER = AtomicRefCounter> virtual decaf::lang::Pointer< T, REFCOUNTER >::~Pointer () throw () [inline, virtual]

6.410.4 Member Function Documentation

6.410.4.1 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::dynamicCast () const [inline]

6.410.4.2 template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::get () const [inline]

Gets the real pointer that is contained within this **Pointer** (p. 2011). This is not really safe since the caller could delete or alter the pointer but it mimics the STL auto_ptr and gives access in cases where the caller absolutely needs the real **Pointer** (p. 2011). Use at your own risk.

Returns:

the contained pointer.

Referenced by activemq::core::ActiveMQConnectionSupport::getProperties(), activemq::core::ActiveMQConnectionSupport::getTransport(), decaf::lang::operator!=(), decaf::lang::Pointer< TransactionId >::operator!=(), std::less< decaf::lang::Pointer< T > >::operator()(), decaf::lang::operator==(), decaf::lang::Pointer< TransactionId >::operator==(), and activemq::state::ConnectionState::removeTempDestination().

6.410.4.3 template<typename T, typename REFCOUNTER = AtomicRefCounter> bool decaf::lang::Pointer< T, REFCOUNTER >::operator! () const [inline]

6.410.4.4 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator!= (const Pointer< T1, R1 > & *right*) const [inline]

6.410.4.5 template<typename T, typename REFCOUNTER = AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator* () const [inline]

6.410.4.6 template<typename T, typename REFCOUNTER = AtomicRefCounter> ReferenceType decaf::lang::Pointer< T, REFCOUNTER >::operator* () [inline]

Dereference Operator, returns a reference to the Contained value. This method throws an NullPointerException if the contained value is NULL.

Returns:

reference to the contained pointer.

Exceptions:

NullPointerException if the contained value is Null

6.410.4.7 template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () const [inline]

6.410.4.8 template<typename T, typename REFCOUNTER = AtomicRefCounter> PointerType decaf::lang::Pointer< T, REFCOUNTER >::operator-> () [inline]

Indirection Operator, returns a pointer to the Contained value. This method throws an NullPointerException if the contained value is NULL.

Returns:

reference to the contained pointer.

Exceptions:

NullPointerException if the contained value is Null

6.410.4.9 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T1, R1 > & *right*) throw () [inline]

6.410.4.10 template<typename T, typename REFCOUNTER = AtomicRefCounter> Pointer& decaf::lang::Pointer< T, REFCOUNTER >::operator= (const Pointer< T, REFCOUNTER > & *right*) throw () [inline]

Assigns the value of right to this **Pointer** (p. 2011) and increments the reference Count.

Parameters:

right - **Pointer** (p. 2011) on the right hand side of an operator= call to this.

6.410.4.11 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 , typename R1 > bool decaf::lang::Pointer< T, REFCOUNTER >::operator== (const Pointer< T1, R1 > & *right*) const [inline]

6.410.4.12 template<typename T, typename REFCOUNTER = AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::reset (T * *value*) [inline]

Resets the **Pointer** (p. 2011) to hold the new value. Before the new value is stored reset checks if the old value should be destroyed and if so calls delete. Call reset with a value of NULL is supported and acts to set this **Pointer** (p. 2011) to a NULL pointer.

Parameters:

value - The new value to contain.

6.410.4.13 template<typename T, typename REFCOUNTER = AtomicRefCounter> template<typename T1 > Pointer<T1, CounterType> decaf::lang::Pointer< T, REFCOUNTER >::staticCast () const [inline]

6.410.4.14 template<typename T, typename REFCOUNTER = AtomicRefCounter> void decaf::lang::Pointer< T, REFCOUNTER >::swap (Pointer< T, REFCOUNTER > & *value*) throw () [inline]

Exception (p. 1268) Safe Swap Function.

Parameters:

value - the value to swap with this.

Referenced by decaf::lang::Pointer< TransactionId >::operator=(), and decaf::lang::Pointer< TransactionId >::swap().

6.410.5 Friends And Related Function Documentation

6.410.5.1 template<typename T, typename REFCOUNTER = AtomicRefCounter> bool operator!= (const T * *left*, const Pointer< T, REFCOUNTER > & *right*) [friend]

6.410.5.2 template<typename T, typename REFCOUNTER = AtomicRefCounter> bool operator!= (const Pointer< T, REFCOUNTER > & *left*, const T * *right*) [friend]

6.410.5.3 template<typename T, typename REFCOUNTER = AtomicRefCounter> bool operator== (const T * *left*, const Pointer< T, REFCOUNTER > & *right*) [friend]

6.410.5.4 template<typename T, typename REFCOUNTER = AtomicRefCounter> bool operator== (const Pointer< T, REFCOUNTER > & *left*, const T * *right*) [friend]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.411 decaf::lang::PointerComparator< T, R > Class Template Reference

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2011) instance.

`#include <src/main/decaf/lang/Pointer.h>`Inheritance diagram for decaf::lang::PointerComparator< T, R >:

Public Member Functions

- virtual bool **operator()** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const
- virtual int **compare** (const **Pointer**< T, R > &left, const **Pointer**< T, R > &right) const

6.411.1 Detailed Description

template<typename T, typename R = AtomicRefCounter> class decaf::lang::PointerComparator< T, R >

This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2011) instance. This can be useful in the case where a series of values in a Collection is more efficiently accessed in the Objects Natural Order and not the underlying pointers location in memory.

Also this allows **Pointer** (p. 2011) objects that Point to different instances of the same type to be compared based on the comparison of the object itself and not just the value of the pointer.

6.411.2 Member Function Documentation

6.411.2.1 template<typename T , typename R = AtomicRefCounter> virtual int decaf::lang::PointerComparator< T, R >::compare (const Pointer< T, R > & *left*, const Pointer< T, R > & *right*) const [inline, virtual]

6.411.2.2 template<typename T , typename R = AtomicRefCounter> virtual bool decaf::lang::PointerComparator< T, R >::operator() (const Pointer< T, R > & *left*, const Pointer< T, R > & *right*) const [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.412 activemq::cmsutil::PooledSession Class Reference

A pooled session object that wraps around a delegate session.

`#include <src/main/activemq/cmsutil/PooledSession.h>`Inheritance diagram for activemq::cmsutil::PooledSession:

Public Member Functions

- **PooledSession** (**SessionPool** *pool, **cms::Session** *session)
- virtual ~**PooledSession** ()

  Does nothing.

- virtual **cms::Session** * **getSession** ()

  Returns a non-constant reference to the internal session object.

- virtual const **cms::Session** * **getSession** () const

  Returns a constant reference to the internal session object.

- virtual void **close** () throw (cms::CMSException)

  Returns this session back to the pool, but does not close or destroy the internal session object.

- virtual void **commit** () throw (cms::CMSException)

  Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** () throw (cms::CMSException)

  Rolls back all messages done in this transaction and releases any locks currently held.

- virtual void **recover** () throw (cms::CMSException)

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSException)

  Creates a MessageConsumer for the specified destination, using a message selector.

- virtual **cms::MessageConsumer** * **createDurableConsumer** (const **cms::Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false) throw (cms::CMSException)

  Creates a durable subscriber to the specified topic, using a Message selector.

- virtual **cms::MessageConsumer** * **createCachedConsumer** (const **cms::Destination** *destination, const std::string &selector, bool noLocal) throw (cms::CMSException)

  First checks the internal consumer cache and creates on if none exist for the given destination, selector, noLocal.

- virtual **cms::MessageProducer** * **createProducer** (const **cms::Destination** *destination) throw (cms::CMSException)

  Creates a MessageProducer to send messages to the specified destination.

- virtual **cms::MessageProducer** * **createCachedProducer** (const **cms::Destination** *destination) throw (cms::CMSException)

  First checks the internal producer cache and creates one if none exist for the given destination.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue) throw (cms::CMSException)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector) throw (cms::CMSException)

  Creates a new QueueBrowser to peek at Messages on the given Queue.

- virtual **cms::Queue** * **createQueue** (const std::string &queueName) throw (cms::CMSException)

  Creates a queue identity given a Queue name.

- virtual **cms::Topic** * **createTopic** (const std::string &topicName) throw (cms::CMSException)

  Creates a topic identity given a Queue name.

- virtual **cms::TemporaryQueue** * **createTemporaryQueue** () throw (cms::CMSException)

  Creates a TemporaryQueue object.

- virtual **cms::TemporaryTopic** * **createTemporaryTopic** () throw (cms::CMSException)

  Creates a TemporaryTopic object.

- virtual **cms::Message** * **createMessage** () throw (cms::CMSException)

  Creates a new Message.

- virtual **cms::BytesMessage** * **createBytesMessage** () throw (cms::CMSException)

  Creates a BytesMessage.

- virtual **cms::BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize) throw (cms::CMSException)

  Creates a BytesMessage and sets the payload to the passed value.

- virtual **cms::StreamMessage** * **createStreamMessage** () throw (cms::CMSException)

  Creates a new StreamMessage.

- virtual **cms::TextMessage** * **createTextMessage** () throw (cms::CMSException)

  Creates a new TextMessage.

- virtual **cms::TextMessage** * **createTextMessage** (const std::string &text) throw (cms::CMSException)

  Creates a new TextMessage and set the text to the value given.

- virtual **cms::MapMessage** * **createMapMessage** () throw (cms::CMSException)

  Creates a new MapMessage.

- virtual **cms::Session::AcknowledgeMode** **getAcknowledgeMode** () const throw (cms::CMSException)

  Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () const throw (cms::CMSException)

  Gets if the Sessions is a Transacted Session.

- virtual void **unsubscribe** (const std::string &name) throw (cms::CMSException)

  Unsubscribes a durable subscription that has been created by a client.

6.412.1 Detailed Description

A pooled session object that wraps around a delegate session. Calls to close this session only result in giving the session back to the pool.

6.412.2 Constructor & Destructor Documentation

6.412.2.1 activemq::cmsutil::PooledSession::PooledSession (SessionPool * *pool*, cms::Session * *session*)

6.412.2.2 virtual activemq::cmsutil::PooledSession::~PooledSession () [virtual]

Does nothing.

6.412.3 Member Function Documentation

6.412.3.1 virtual void activemq::cmsutil::PooledSession::close () throw (cms::CMSException) [virtual]

Returns this session back to the pool, but does not close or destroy the internal session object.

Implements **cms::Session** (p. 2273).

6.412.3.2 virtual void activemq::cmsutil::PooledSession::commit () throw (cms::CMSException) [inline, virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements **cms::Session** (p. 2274).

References cms::Session::commit().

6.412.3.3 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue*, const std::string & *selector*) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

selector the Message selector to filter which messages are browsed.

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2274).

6.412.3.4 virtual cms::QueueBrowser* activemq::cmsutil::PooledSession::createBrowser (const cms::Queue * *queue*) throw (cms::CMSException) [virtual]

Creates a new QueueBrowser to peek at Messages on the given Queue.

Parameters:

queue the Queue to browse

Returns:

New QueueBrowser that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if the destination given is invalid.

Implements **cms::Session** (p. 2274).

6.412.3.5 virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage (const unsigned char * *bytes*, std::size_t *bytesSize*) throw (cms::CMSException) [inline, virtual]

Creates a BytesMessage and sets the payload to the passed value.

Parameters:

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2275).

References cms::Session::createBytesMessage().

6.412.3.6 virtual cms::BytesMessage* activemq::cmsutil::PooledSession::createBytesMessage () throw (cms::CMSException) [inline, virtual]

Creates a BytesMessage.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2275).

References cms::Session::createBytesMessage().

6.412.3.7 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createCachedConsumer (const cms::Destination * *destination*, const std::string & *selector*, bool *noLocal*) throw (cms::CMSException) [virtual]

First checks the internal consumer cache and creates on if none exist for the given destination, selector, noLocal. If created, the consumer is added to the pool's lifecycle manager.

Parameters:

destination the destination to receive on

selector the selector to use

noLocal whether or not to receive messages from the same connection

Returns:

the consumer resource

Exceptions:

cms::CMSException (p. ***850***) if something goes wrong.

6.412.3.8 virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createCachedProducer (const cms::Destination * *destination*) throw (cms::CMSException) [virtual]

First checks the internal producer cache and creates one if none exist for the given destination. If created, the producer is added to the pool's lifecycle manager.

Parameters:

destination the destination to send on

Returns:

the producer resource

Exceptions:

cms::CMSException *(*p. ***850)*** if something goes wrong.

6.412.3.9 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*, bool *noLocal*) throw (cms::CMSException) [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2275).

References cms::Session::createConsumer().

6.412.3.10 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*, const std::string & *selector*) throw (cms::CMSException) [inline, virtual]

Creates a MessageConsumer for the specified destination, using a message selector.

Parameters:

destination the Destination that this consumer receiving messages for.

selector the Message Selector to use

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2276).

References cms::Session::createConsumer().

6.412.3.11 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createConsumer (const cms::Destination * *destination*) throw (cms::CMSException) [inline, virtual]

Creates a MessageConsumer for the specified destination.

Parameters:

destination the Destination that this consumer receiving messages for.

Returns:

pointer to a new MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2276).

References cms::Session::createConsumer().

6.412.3.12 virtual cms::MessageConsumer* activemq::cmsutil::PooledSession::createDurableConsumer (const cms::Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = false) throw (cms::CMSException) [inline, virtual]

Creates a durable subscriber to the specified topic, using a Message selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters:

destination the topic to subscribe to

name The name used to identify the subscription

selector the Message Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new durable MessageConsumer that is owned by the caller (caller deletes)

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

InvalidSelectorException - if the message selector is invalid.

Implements **cms::Session** (p. 2277).

References cms::Session::createDurableConsumer().

6.412.3.13 virtual cms::MapMessage* activemq::cmsutil::PooledSession::createMapMessage () throw (cms::CMSException) [inline, virtual]

Creates a new MapMessage.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2277).

References cms::Session::createMapMessage().

6.412.3.14 virtual cms::Message* activemq::cmsutil::PooledSession::createMessage () throw (cms::CMSException) [inline, virtual]

Creates a new Message.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2278).

References cms::Session::createMessage().

6.412.3.15 virtual cms::MessageProducer* activemq::cmsutil::PooledSession::createProducer (const cms::Destination * *destination*) throw (cms::CMSException) [inline, virtual]

Creates a MessageProducer to send messages to the specified destination.

Parameters:

destination the Destination to send on

Returns:

New MessageProducer that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

InvalidDestinationException - if an invalid destination is specified.

Implements **cms::Session** (p. 2278).

References cms::Session::createProducer().

6.412.3.16 virtual cms::Queue* activemq::cmsutil::PooledSession::createQueue (const std::string & *queueName*) throw (cms::CMSException) [inline, virtual]

Creates a queue identity given a Queue name.

Parameters:

queueName the name of the new Queue

Returns:

new Queue pointer that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2278).

References cms::Session::createQueue().

6.412.3.17 virtual cms::StreamMessage* activemq::cmsutil::PooledSession::createStreamMessage () throw (cms::CMSException) [inline, virtual]

Creates a new StreamMessage.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2279).

References cms::Session::createStreamMessage().

6.412.3.18 virtual cms::TemporaryQueue* activemq::cmsutil::PooledSession::createTemporaryQueue () throw (cms::CMSException) [inline, virtual]

Creates a TemporaryQueue object.

Returns:

new TemporaryQueue pointer that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2279).

References cms::Session::createTemporaryQueue().

6.412.3.19 virtual cms::TemporaryTopic* activemq::cmsutil::PooledSession::createTemporaryTopic () throw (cms::CMSException) [inline, virtual]

Creates a TemporaryTopic object.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2279).

References cms::Session::createTemporaryTopic().

6.412.3.20 virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage (const std::string & *text*) throw (cms::CMSException) [inline, virtual]

Creates a new TextMessage and set the text to the value given.

Parameters:

text the initial text for the message

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2279).

References cms::Session::createTextMessage().

6.412.3.21 virtual cms::TextMessage* activemq::cmsutil::PooledSession::createTextMessage () throw (cms::CMSException) [inline, virtual]

Creates a new TextMessage.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2280).

References cms::Session::createTextMessage().

6.412.3.22 virtual cms::Topic* activemq::cmsutil::PooledSession::createTopic (const std::string & *topicName*) throw (cms::CMSException) [inline, virtual]

Creates a topic identity given a Queue name.

Parameters:

topicName the name of the new Topic

Returns:

new Topic pointer that is owned by the caller.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2280).

References cms::Session::createTopic().

6.412.3.23 virtual cms::Session::AcknowledgeMode activemq::cmsutil::PooledSession::getAcknowledgeMode () const throw (cms::CMSException) [inline, virtual]

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2280).

References cms::Session::getAcknowledgeMode().

6.412.3.24 virtual const cms::Session* activemq::cmsutil::PooledSession::getSession () const [inline, virtual]

Returns a constant reference to the internal session object.

Returns:

the session object.

6.412.3.25 virtual cms::Session* activemq::cmsutil::PooledSession::getSession () [inline, virtual]

Returns a non-constant reference to the internal session object.

Returns:

the session object.

6.412.3.26 virtual bool activemq::cmsutil::PooledSession::isTransacted () const throw (cms::CMSException) [inline, virtual]

Gets if the Sessions is a Transacted Session.

Returns:

transacted true - false.

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2281).

References cms::Session::isTransacted().

6.412.3.27 virtual void activemq::cmsutil::PooledSession::recover () throw (cms::CMSException) [inline, virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSException - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException - if the method is called by a transacted session.

Implements **cms::Session** (p. 2281).

References cms::Session::recover().

6.412.3.28 virtual void activemq::cmsutil::PooledSession::rollback () throw (cms::CMSException) [inline, virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException - If an internal error occurs.

IllegalStateException - if the method is not called by a transacted session.

Implements **cms::Session** (p. 2281).

References cms::Session::rollback().

6.412.3.29 virtual void activemq::cmsutil::PooledSession::unsubscribe (const std::string & *name*) throw (cms::CMSException) [inline, virtual]

Unsubscribes a durable subscription that has been created by a client. This method deletes the **state** (p. 65) being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active MessageConsumer or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name The name used to identify this subscription

Exceptions:

CMSException - If an internal error occurs.

Implements **cms::Session** (p. 2282).

References cms::Session::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**PooledSession.h**

6.413 decaf::util::concurrent::PooledThread Class Reference

`#include <src/main/decaf/util/concurrent/PooledThread.h>`Inheritance diagram for decaf::util::concurrent::PooledThread:

Public Member Functions

- **PooledThread** (**ThreadPool** *pool)

  Constructor.

- virtual ~**PooledThread** ()
- virtual void **run** ()

  *Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 2549) and then grabs it and calls its run method.*

- virtual void **stop** () throw (lang::Exception)

  Stops the Thread, thread will complete its task if currently running one, and then die.

- virtual bool **isBusy** ()

  Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.

- virtual void **setPooledThreadListener** (**PooledThreadListener** *listener)

  Adds a listener to this `PooledThread` (p. 2032) to be notified when this thread starts and completes a task.

- virtual **PooledThreadListener** * **getPooledThreadListener** ()

  Removes a listener for this `PooledThread` (p. 2032) to be notified when this thread starts and completes a task.

6.413.1 Constructor & Destructor Documentation

6.413.1.1 decaf::util::concurrent::PooledThread::PooledThread (ThreadPool * *pool*)

Constructor.

Parameters:

pool the parant **ThreadPool** (p. 2549) object

6.413.1.2 virtual decaf::util::concurrent::PooledThread::~PooledThread () [virtual]

6.413.2 Member Function Documentation

6.413.2.1 virtual PooledThreadListener* decaf::util::concurrent::PooledThread::getPooledThreadListener () [inline, virtual]

Removes a listener for this PooledThread (p. 2032) to be notified when this thread starts and completes a task.

Returns:

a pointer to this thread's listener or NULL

6.413.2.2 virtual bool decaf::util::concurrent::PooledThread::isBusy () [inline, virtual]

Checks to see if the thread is busy, if busy it means that this thread has taken a task from the ThreadPool's queue and is processing it.

Returns:

true if the Thread is busy

6.413.2.3 virtual void decaf::util::concurrent::PooledThread::run () [virtual]

Run Method for this object waits for something to be enqueued on the **ThreadPool** (p. 2549) and then grabs it and calls its run method.

Reimplemented from **decaf::lang::Thread** (p. 2545).

6.413.2.4 virtual void decaf::util::concurrent::PooledThread::setPooledThreadListener (PooledThreadListener * *listener*) [inline, virtual]

Adds a listener to this PooledThread (p. 2032) to be notified when this thread starts and completes a task.

Parameters:

listener the listener to send notifications to.

6.413.2.5 virtual void decaf::util::concurrent::PooledThread::stop () throw (lang::Exception) [virtual]

Stops the Thread, thread will complete its task if currently running one, and then die. Does not block.

Exceptions:

Exception

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThread.h**

6.414 decaf::util::concurrent::PooledThreadListener Class Reference

Abstract Listener Interface for users of `ThreadPool` (p. 2549).

`#include <src/main/decaf/util/concurrent/PooledThreadListener.h>`Inheritance diagram for decaf::util::concurrent::PooledThreadListener:

Public Member Functions

- virtual ~**PooledThreadListener** ()
- virtual void **onTaskStarted** (**PooledThread** *thread)=0

  Called by a pooled thread when it is about to begin executing a new task.

- virtual void **onTaskCompleted** (**PooledThread** *thread)=0

  Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

- virtual void **onTaskException** (**PooledThread** *thread, **lang::Exception** &ex)=0

  *Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2032) is now no longer running.*

6.414.1 Detailed Description

Abstract Listener Interface for users of `ThreadPool` (p. 2549). The implementor of this class receives events related to the execution and termination of threads running in the **ThreadPool** (p. 2549).

Since:

1.0

6.414.2 Constructor & Destructor Documentation

6.414.2.1 virtual decaf::util::concurrent::PooledThreadListener::~PooledThreadListener () [inline, virtual]

6.414.3 Member Function Documentation

6.414.3.1 virtual void decaf::util::concurrent::PooledThreadListener::onTaskCompleted (PooledThread * *thread*) [pure virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run.

Parameters:

thread - Pointer the the Pooled Thread that is making this call.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 2552).

6.414.3.2 virtual void decaf::util::concurrent::PooledThreadListener::onTaskException (PooledThread * *thread*, lang::Exception & *ex*) [pure virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2032) is now no longer running.

Parameters:

thread - Pointer to the Pooled Thread that is making this call

ex - The Exception that occurred.

Implemented in **decaf::util::concurrent::ThreadPool** (p. 2552).

6.414.3.3 virtual void decaf::util::concurrent::PooledThreadListener::onTaskStarted (PooledThread * *thread*) [pure virtual]

Called by a pooled thread when it is about to begin executing a new task.

Parameters:

thread - Pointer to the Pooled Thread that is making this call

Implemented in **decaf::util::concurrent::ThreadPool** (p. 2553).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**PooledThreadListener.h**

6.415 decaf::net::PortUnreachableException Class Reference

`#include <src/main/decaf/net/PortUnreachableException.h>`Inheritance diagram for decaf::net::PortUnreachableException:

Public Member Functions

- **PortUnreachableException** () throw ()

  Default Constructor.

- **PortUnreachableException** (const Exception &ex) throw ()

  Conversion Constructor from some other Exception.

- **PortUnreachableException** (const **PortUnreachableException** &ex) throw ()

  Copy Constructor.

- **PortUnreachableException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **PortUnreachableException** (const std::exception ***cause**) throw ()

  Constructor.

- **PortUnreachableException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual **PortUnreachableException** * **clone** () const

  Clones this exception.

- virtual ~**PortUnreachableException** () throw ()

6.415.1 Constructor & Destructor Documentation

6.415.1.1 decaf::net::PortUnreachableException::PortUnreachableException () throw () [inline]

Default Constructor.

6.415.1.2 decaf::net::PortUnreachableException::PortUnreachableException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.415.1.3 decaf::net::PortUnreachableException::PortUnreachableException (const PortUnreachableException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.415.1.4 decaf::net::PortUnreachableException::PortUnreachableException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.415.1.5 decaf::net::PortUnreachableException::PortUnreachableException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.415.1.6 decaf::net::PortUnreachableException::PortUnreachableException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.415.1.7 virtual decaf::net::PortUnreachableException::~PortUnreachableException () throw () [inline, virtual]

6.415.2 Member Function Documentation

6.415.2.1 virtual PortUnreachableException* decaf::net::PortUnreachableException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::net::SocketException** (p. 2380).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**PortUnreachableException.h**

6.416 activemq::util::PrimitiveList Class Reference

List of primitives.

`#include <src/main/activemq/util/PrimitiveList.h>`Inheritance diagram for activemq::util::PrimitiveList:

Public Member Functions

- **PrimitiveList** ()

  Default Constructor, creates an Empty list.

- virtual ~**PrimitiveList** ()
- **PrimitiveList** (const **decaf::util::List**< **PrimitiveValueNode** > &src)

  Copy Constructor.

- **PrimitiveList** (const **PrimitiveList** &src)

  Copy Constructor.

- std::string **toString** () const

  Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

- virtual bool **getBool** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Boolean value at the specified index.

- virtual void **setBool** (std::size_t index, bool value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual unsigned char **getByte** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Byte value at the specified index.

- virtual void **setByte** (std::size_t index, unsigned char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual char **getChar** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Character value at the specified index.

- virtual void **setChar** (std::size_t index, char value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual short **getShort** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Short value at the specified index.

- virtual void **setShort** (std::size_t index, short value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual int **getInt** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Integer value at the specified index.

- virtual void **setInt** (std::size_t index, int value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual long long **getLong** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Long value at the specified index.

- virtual void **setLong** (std::size_t index, long long value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual float **getFloat** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Float value at the specified index.

- virtual void **setFloat** (std::size_t index, float value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual double **getDouble** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

Gets the Double value at the specified index.

- virtual void **setDouble** (std::size_t index, double value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::string **getString** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the String value at the specified index.

- virtual void **setString** (std::size_t index, const std::string &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

- virtual std::vector< unsigned char > **getByteArray** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Byte Array value at the specified index.

- virtual void **setByteArray** (std::size_t index, const std::vector< unsigned char > &value) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

6.416.1 Detailed Description

List of primitives.

6.416.2 Constructor & Destructor Documentation

6.416.2.1 activemq::util::PrimitiveList::PrimitiveList ()

Default Constructor, creates an Empty list.

6.416.2.2 virtual activemq::util::PrimitiveList::~PrimitiveList () [virtual]

6.416.2.3 activemq::util::PrimitiveList::PrimitiveList (const decaf::util::List< PrimitiveValueNode > & *src*)

Copy Constructor.

Parameters:

src - the Decaf List of PrimtiveNodeValues to copy

6.416.2.4 activemq::util::PrimitiveList::PrimitiveList (const PrimitiveList & *src*)

Copy Constructor.

Parameters:

src - the **PrimitiveList** (p. 2040) to copy

6.416.3 Member Function Documentation

6.416.3.1 virtual bool activemq::util::PrimitiveList::getBool (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Boolean value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.2 virtual unsigned char activemq::util::PrimitiveList::getByte (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveList::getByteArray (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte Array value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.4 virtual char activemq::util::PrimitiveList::getChar (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Character value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.5 virtual double activemq::util::PrimitiveList::getDouble (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Double value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.6 virtual float activemq::util::PrimitiveList::getFloat (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Float value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.7 virtual int activemq::util::PrimitiveList::getInt (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Integer value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.8 virtual long long activemq::util::PrimitiveList::getLong (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Long value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.9 virtual short activemq::util::PrimitiveList::getShort (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Short value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.10 virtual std::string activemq::util::PrimitiveList::getString (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the String value at the specified index.

Parameters:

index - index to get value from

Returns:

value contained at the given index

Exceptions:

IndexOutOfBoundsException if index is > **size()** (p. 2428)

UnsupportedOperationException if the type at index is not of the type that this method is to return or can convert to.

6.416.3.11 virtual void activemq::util::PrimitiveList::setBool (std::size_t *index*, bool *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.12 virtual void activemq::util::PrimitiveList::setByte (std::size_t *index*, unsigned char *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.13 virtual void activemq::util::PrimitiveList::setByteArray (std::size_t *index*, const std::vector< unsigned char > & *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.14 virtual void activemq::util::PrimitiveList::setChar (std::size_t *index*, char *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.15 virtual void activemq::util::PrimitiveList::setDouble (std::size_t *index*, double *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.16 virtual void activemq::util::PrimitiveList::setFloat (std::size_t *index*, float *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.17 virtual void activemq::util::PrimitiveList::setInt (std::size_t *index*, int *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.18 virtual void activemq::util::PrimitiveList::setLong (std::size_t *index*, long long *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.19 virtual void activemq::util::PrimitiveList::setShort (std::size_t *index*, short *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.20 virtual void activemq::util::PrimitiveList::setString (std::size_t *index*, const std::string & *value*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [virtual]

Sets the value at the given index to the new value specified, this method overwrites any data that was previously at the index given, but does not insert a new element if the index is greater than the size of the list.

Parameters:

index - location to set in the list

value - the new value to assign to the element at index

Exceptions:

IndexOutOfBoundsException if index > **size()** (p. 2428).

6.416.3.21 std::string activemq::util::PrimitiveList::toString () const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns:

formatted String of all elements in the list.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveList.h**

6.417 activemq::util::PrimitiveMap Class Reference

Map of named primitives.

`#include <src/main/activemq/util/PrimitiveMap.h>`Inheritance diagram for activemq::util::PrimitiveMap:

Public Member Functions

- **PrimitiveMap** ()

  Default Constructor, creates an empty map.

- virtual ~**PrimitiveMap** ()
- **PrimitiveMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &source)

  Copy Constructor.

- **PrimitiveMap** (const **PrimitiveMap** &source)

  Copy Constructor.

- std::string **toString** () const

  Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

- virtual bool **getBool** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setBool** (const std::string &key, bool value)

  Sets the value at key to the specified type.

- virtual unsigned char **getByte** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setByte** (const std::string &key, unsigned char value)

  Sets the value at key to the specified type.

- virtual char **getChar** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setChar** (const std::string &key, char value)

Sets the value at key to the specified type.

- virtual short **getShort** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setShort** (const std::string &key, short value)

  Sets the value at key to the specified type.

- virtual int **getInt** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setInt** (const std::string &key, int value)

  Sets the value at key to the specified type.

- virtual long long **getLong** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setLong** (const std::string &key, long long value)

  Sets the value at key to the specified type.

- virtual float **getFloat** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setFloat** (const std::string &key, float value)

  Sets the value at key to the specified type.

- virtual double **getDouble** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setDouble** (const std::string &key, double value)

  Sets the value at key to the specified type.

- virtual std::string **getString** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setString** (const std::string &key, const std::string &value)

  Sets the value at key to the specified type.

- virtual std::vector< unsigned char > **getByteArray** (const std::string &key) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

- virtual void **setByteArray** (const std::string &key, const std::vector< unsigned char > &value)

  Sets the value at key to the specified type.

6.417.1 Detailed Description

Map of named primitives.

6.417.2 Constructor & Destructor Documentation

6.417.2.1 activemq::util::PrimitiveMap::PrimitiveMap ()

Default Constructor, creates an empty map.

6.417.2.2 virtual activemq::util::PrimitiveMap::~PrimitiveMap () [virtual]

6.417.2.3 activemq::util::PrimitiveMap::PrimitiveMap (const decaf::util::Map< std::string, PrimitiveValueNode > & *source*)

Copy Constructor.

Parameters:

source The Decaf Library Map instance whose elements will be copied into this Map.

6.417.2.4 activemq::util::PrimitiveMap::PrimitiveMap (const PrimitiveMap & *source*)

Copy Constructor.

Parameters:

source The **PrimitiveMap** (p. 2051) whose elements will be copied into this Map.

6.417.3 Member Function Documentation

6.417.3.1 virtual bool activemq::util::PrimitiveMap::getBool (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Boolean value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.2 virtual unsigned char activemq::util::PrimitiveMap::getByte (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.3 virtual std::vector<unsigned char> activemq::util::PrimitiveMap::getByteArray (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Byte Array value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.4 virtual char activemq::util::PrimitiveMap::getChar (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Character value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.5 virtual double activemq::util::PrimitiveMap::getDouble (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Double value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.6 virtual float activemq::util::PrimitiveMap::getFloat (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Float value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.7 virtual int activemq::util::PrimitiveMap::getInt (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Integer value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.8 virtual long long activemq::util::PrimitiveMap::getLong (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Long value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.9 virtual short activemq::util::PrimitiveMap::getShort (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the Short value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.10 virtual std::string activemq::util::PrimitiveMap::getString (const std::string & *key*) const throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Gets the String value at the given key, if the key is not in the map or cannot be returned as the requested value then an exception of type NoSuchElementException is thrown.

Parameters:

key - the location to return the value from.

Returns:

the value at key in the type requested.

Exceptions:

NoSuchElementException if key is not in the map.

UnSupportedOperationException if the value cannot be converted to the type this method returns

6.417.3.11 virtual void activemq::util::PrimitiveMap::setBool (const std::string & *key*, bool *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.12 virtual void activemq::util::PrimitiveMap::setByte (const std::string & *key*, unsigned char *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.13 virtual void activemq::util::PrimitiveMap::setByteArray (const std::string & *key*, const std::vector< unsigned char > & *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.14 virtual void activemq::util::PrimitiveMap::setChar (const std::string & *key*, char *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.15 virtual void activemq::util::PrimitiveMap::setDouble (const std::string & *key*, double *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.16 virtual void activemq::util::PrimitiveMap::setFloat (const std::string & *key*, float *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.17 virtual void activemq::util::PrimitiveMap::setInt (const std::string & *key*, int *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.18 virtual void activemq::util::PrimitiveMap::setLong (const std::string & *key*, long long *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.19 virtual void activemq::util::PrimitiveMap::setShort (const std::string & *key*, short *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.20 virtual void activemq::util::PrimitiveMap::setString (const std::string & *key*, const std::string & *value*) [virtual]

Sets the value at key to the specified type. Overwrites any data that was previously at this key or inserts a new element at key.

Parameters:

key - the map key to set or insert.

value - the new value to set at the key location.

6.417.3.21 std::string activemq::util::PrimitiveMap::toString () const

Converts the contents into a formatted string that can be output in a Log File or other debugging tool.

Returns:

formatted String of all elements in the map.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveMap.h**

6.418 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshall Class Reference

This class wraps the functionality needed to **marshal** (p. 76) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

`#include <src/main/activemq/wireformat/openwire/marshal/PrimitiveTypesMarshaller.h>`

Public Member Functions

- **PrimitiveTypesMarshaller** ()
- virtual ~**PrimitiveTypesMarshaller** ()

Static Public Member Functions

- static void **marshal** (const **util::PrimitiveMap** *map, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)

  Static Marshal of a primitive map object.

- static void **unmarshal** (**util::PrimitiveMap** *map, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)

  Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

- static void **marshal** (const **util::PrimitiveList** *list, std::vector< unsigned char > &dest) throw (decaf::lang::Exception)

  Static Marshal of a primitive map object.

- static void **unmarshal** (**util::PrimitiveList** *list, const std::vector< unsigned char > &src) throw (decaf::lang::Exception)

  Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object.

Static Protected Member Functions

- static void **marshalPrimitiveMap** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::Map**< std::string, **util::PrimitiveValueNode** > &map) throw (decaf::io::IOException)

  Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

- static void **marshalPrimitiveList** (**decaf::io::DataOutputStream** &dataOut, const **decaf::util::List**< **util::PrimitiveValueNode** > &list) throw (decaf::io::IOException)

  Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

- static void **marshalPrimitive** (**decaf::io::DataOutputStream** &dataOut, const **util::PrimitiveValueNode** &value) throw (decaf::io::IOException)

  Used to Marshal the Primitive types out on the Wire.

- static void **unmarshalPrimitiveMap** (**decaf::io::DataInputStream** &dataIn, **util::PrimitiveMap** &map) throw (decaf::io::IOException)

  Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

- static void **unmarshalPrimitiveList** (**decaf::io::DataInputStream** &dataIn, **decaf::util::StlList**< **util::PrimitiveValueNode** > &list) throw (decaf::io::IOException)

  Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

- static **util::PrimitiveValueNode unmarshalPrimitive** (**decaf::io::DataInputStream** &dataIn) throw (decaf::io::IOException)

  Unmarshals a Primitive Type from the stream, and returns it as a value Node.

6.418.1 Detailed Description

This class wraps the functionality needed to **marshal** (p. 76) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.

6.418.2 Constructor & Destructor Documentation

6.418.2.1 activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::PrimitiveTypesMars () [inline]

6.418.2.2 virtual activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::~PrimitiveTypesMa () [inline, virtual]

6.418.3 Member Function Documentation

6.418.3.1 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveList * *list*, std::vector< unsigned char > & *dest*) throw (decaf::lang::Exception) [static]

Static Marshal of a primitive map object.

Parameters:

list The list object to Marshal

dest Reference to a byte array to house the data

Exceptions:

Exception

6.418.3.2 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshal (const util::PrimitiveMap * *map*, std::vector< unsigned char > & *dest*) throw (decaf::lang::Exception) [static]

Static Marshal of a primitive map object.

Parameters:

map Map to Marshal.

dest Reference to a byte array to house the data.

Exceptions:

Exception

6.418.3.3 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitive (decaf::io::DataOutputStream & *dataOut*, const util::PrimitiveValueNode & *value*) throw (decaf::io::IOException) [static, protected]

Used to Marshal the Primitive types out on the Wire.

Parameters:

dataOut - the DataOutputStream to write to

value - the ValueNode to write.

Exceptions:

IOException

6.418.3.4 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveList (decaf::io::DataOutputStream & *dataOut*, const decaf::util::List< util::PrimitiveValueNode > & *list*) throw (decaf::io::IOException) [static, protected]

Marshal a List of Primitives to the given OutputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters:

dataOut - the DataOutputStream to write to

list - the ValueNode to write.

Exceptions:

IOException

6.418.3.5 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::marshalPrimitiveMap (decaf::io::DataOutputStream & *dataOut*, const decaf::util::Map< std::string, util::PrimitiveValueNode > & *map*) throw (decaf::io::IOException) [static, protected]

Marshal a Map of Primitives to the given OutputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters:

dataOut - the DataOutputStream to write to

map - the ValueNode to write.

Exceptions:

IOException

6.418.3.6 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveList * *list*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object. Caller owns the pointer.

Parameters:

list The list object to Un-marshal

src Reference to a byte array to read data from.

Exceptions:

Exception

6.418.3.7 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshal (util::PrimitiveMap * *map*, const std::vector< unsigned char > & *src*) throw (decaf::lang::Exception) [static]

Static Map Unmarshaler, takes an array of bytes and returns a new instance of a PrimitiveMap object. Caller owns the pointer.

Parameters:

map Map to Unmarshal into

src Reference to a byte array to read data from.

Exceptions:

Exception

6.418.3.8 static util::PrimitiveValueNode activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitive (decaf::io::DataInputStream & *dataIn*) throw (decaf::io::IOException) [static, protected]

Unmarshals a Primitive Type from the stream, and returns it as a value Node.

Parameters:

dataIn - DataInputStream to read from.

Returns:

a PrimitiveValueNode containing the data.

Exceptions:

IOException

6.418.3.9 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveLis (decaf::io::DataInputStream & *dataIn*, decaf::util::StlList< util::PrimitiveValueNode > & *list*) throw (decaf::io::IOException) [static, protected]

Unmarshals a List of Primitives from the given InputStream, can result in recursive calls to this method if the list contains lists of lists.

Parameters:

dataIn - DataInputStream to read from.

list - the ValueNode to write.

Exceptions:

IOException

6.418.3.10 static void activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller::unmarshalPrimitiveM (decaf::io::DataInputStream & *dataIn*, util::PrimitiveMap & *map*) throw (decaf::io::IOException) [static, protected]

Unmarshals a Map of Primitives from the given InputStream, can result in recursive calls to this method if the map contains maps of maps.

Parameters:

dataIn - DataInputStream to read from.

map - the map to fill with data.

Exceptions:

IOException

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/**PrimitiveTypesMarshaller.h**

6.419 activemq::util::PrimitiveValueNode::PrimitiveValue Union Reference

Define a union type comprised of the various types.

`#include <src/main/activemq/util/PrimitiveValueNode.h>`

Data Fields

- bool **boolValue**
- unsigned char **byteValue**
- char **charValue**
- short **shortValue**
- int **intValue**
- long long **longValue**
- double **doubleValue**
- float **floatValue**
- std::string * **stringValue**
- std::vector< unsigned char > * **byteArrayValue**
- **decaf::util::List**< **PrimitiveValueNode** > * **listValue**
- **decaf::util::Map**< std::string, **PrimitiveValueNode** > * **mapValue**

6.419.1 Detailed Description

Define a union type comprised of the various types.

6.419.2 Field Documentation

6.419.2.1 bool activemq::util::PrimitiveValueNode::PrimitiveValue::boolValue

6.419.2.2 std::vector<unsigned char>* activemq::util::PrimitiveValueNode::PrimitiveValue::byteArrayValue

6.419.2.3 unsigned char activemq::util::PrimitiveValueNode::PrimitiveValue::byteValue

6.419.2.4 char activemq::util::PrimitiveValueNode::PrimitiveValue::charValue

6.419.2.5 double activemq::util::PrimitiveValueNode::PrimitiveValue::doubleValue

6.419.2.6 float activemq::util::PrimitiveValueNode::PrimitiveValue::floatValue

6.419.2.7 int activemq::util::PrimitiveValueNode::PrimitiveValue::intValue

6.419.2.8 decaf::util::List<PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::listValue

6.419.2.9 long long activemq::util::PrimitiveValueNode::PrimitiveValue::longValue

6.419.2.10 decaf::util::Map<std::string, PrimitiveValueNode>* activemq::util::PrimitiveValueNode::PrimitiveValue::mapValue

6.419.2.11 short activemq::util::PrimitiveValueNode::PrimitiveValue::shortValue

6.419.2.12 std::string* activemq::util::PrimitiveValueNode::PrimitiveValue::stringValue

The documentation for this union was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.420 activemq::util::PrimitiveValueConverter Class Reference

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2070) from one type to another.

`#include <src/main/activemq/util/PrimitiveValueConverter.h>`

Public Member Functions

- **PrimitiveValueConverter** ()
- virtual ~**PrimitiveValueConverter** ()
- template<typename TO >
  TO **convert** (const **PrimitiveValueNode** &value) const throw (decaf::lang::exceptions::UnsupportedOperationException)

6.420.1 Detailed Description

Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2070) from one type to another. If the conversion is supported then calling the convert method will throw an UnsupportedOperationException to indicate that its not possible to perform the conversion.

This class is used to implement the rules of conversion on CMS Message properties, the following conversion table must be implemented. A value written as the row type can be read in the column type.

| | boolean byte short int long float double String |----------|-------|-------|-------|-------|-------|-------|-------| |boolean | X | | | | | | | X | |byte | | X | X | X | X | | | X | |short | | | X | X | X | | | X | |int | | | | X | X | | | X | |long | | | | | X | | | X | |float | | | | | | X | X | X | |double | | | | | | | X | X | |String | X | X | X | X | X | X | X | X |----------|-------|-------|-------|-------|-------|-------|-------|-------|

Since:

3.0

6.420.2 Constructor & Destructor Documentation

6.420.2.1 activemq::util::PrimitiveValueConverter::PrimitiveValueConverter ()

`[inline]`

6.420.2.2 virtual activemq::util::PrimitiveValueConverter::~PrimitiveValueConverter ()

`[inline, virtual]`

6.420.3 Member Function Documentation

6.420.3.1 std::vector< unsigned char > activemq::util::PrimitiveValueConverter::convert (const PrimitiveValueNode & *value*) const throw (decaf::lang::exceptions::UnsupportedOperationException)

`[inline]`

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueConverter.h**

6.421 activemq::util::PrimitiveValueNode Class Reference

Class that wraps around a single value of one of the many types.

`#include <src/main/activemq/util/PrimitiveValueNode.h>`

Data Structures

- union **PrimitiveValue**

  Define a union type comprised of the various types.

Public Types

- enum **PrimitiveType** {

  NULL_TYPE = 0, **BOOLEAN_TYPE** = 1, **BYTE_TYPE** = 2, **CHAR_TYPE** = 3,

  SHORT_TYPE = 4, **INTEGER_TYPE** = 5, **LONG_TYPE** = 6, **DOUBLE_TYPE** = 7,

  FLOAT_TYPE = 8, **STRING_TYPE** = 9, **BYTE_ARRAY_TYPE** = 10, **MAP_TYPE** = 11,

  LIST_TYPE = 12, **BIG_STRING_TYPE** = 13 }

  Enumeration for the various primitive types.

Public Member Functions

- **PrimitiveValueNode** ()

  Default Constructor, creates a value of the NULL_TYPE.

- **PrimitiveValueNode** (bool value)

  Boolean Value Constructor.

- **PrimitiveValueNode** (unsigned char value)

  Byte Value Constructor.

- **PrimitiveValueNode** (char value)

  Char Value Constructor.

- **PrimitiveValueNode** (short value)

  Short Value Constructor.

- **PrimitiveValueNode** (int value)

  Int Value Constructor.

- **PrimitiveValueNode** (long long value)

  Long Value Constructor.

- **PrimitiveValueNode** (float value)

Float Value Constructor.

- **PrimitiveValueNode** (double value)

  Double Value Constructor.

- **PrimitiveValueNode** (const char *value)

  String Value Constructor.

- **PrimitiveValueNode** (const std::string &value)

  String Value Constructor.

- **PrimitiveValueNode** (const std::vector< unsigned char > &value)

  Byte Array Value Constructor.

- **PrimitiveValueNode** (const **decaf::util::List**< **PrimitiveValueNode** > &value)

  Primtive List Constructor.

- **PrimitiveValueNode** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)

  Primtive Map Value Constructor.

- **PrimitiveValueNode** (const **PrimitiveValueNode** &node)

  Copy constructor.

- ~**PrimitiveValueNode** ()
- **PrimitiveValueNode** & **operator**= (const **PrimitiveValueNode** &node)

  Assignment operator, copies the data from the other node.

- bool **operator**== (const **PrimitiveValueNode** &node) const

  Comparison Operator, compares this node to the other node.

- **PrimitiveType getType** () const

  Gets the Value Type of this type wrapper.

- **PrimitiveValue getValue** () const

  Gets the internal Primitive Value object from this wrapper.

- void **setValue** (const **PrimitiveValue** &value, **PrimitiveType** valueType)

  Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

- void **clear** ()

  Clears the value from this wrapper converting it back to a blank NULL_ TYPE value.

- void **setBool** (bool value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- bool **getBool** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Boolean value of this Node.

- void **setByte** (unsigned char value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- unsigned char **getByte** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Byte value of this Node.

- void **setChar** (char value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- char **getChar** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Character value of this Node.

- void **setShort** (short value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- short **getShort** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Short value of this Node.

- void **setInt** (int value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- int **getInt** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Integer value of this Node.

- void **setLong** (long long value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- long long **getLong** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Long value of this Node.

- void **setFloat** (float value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- float **getFloat** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Float value of this Node.

- void **setDouble** (double value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- double **getDouble** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Double value of this Node.

- void **setString** (const std::string &value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- std::string **getString** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the String value of this Node.

- void **setByteArray** (const std::vector< unsigned char > &value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- std::vector< unsigned char > **getByteArray** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Byte Array value of this Node.

- void **setList** (const **decaf::util::List**< **PrimitiveValueNode** > &value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- const **decaf::util::List**< **PrimitiveValueNode** > & **getList** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Primitive List value of this Node.

- void **setMap** (const **decaf::util::Map**< std::string, **PrimitiveValueNode** > &value)

  Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

- const **decaf::util::Map**< std::string, **PrimitiveValueNode** > & **getMap** () const throw (decaf::lang::exceptions::NoSuchElementException)

  Gets the Primitive Map value of this Node.

- std::string **toString** () const

  Creates a string representation of this value.

6.421.1 Detailed Description

Class that wraps around a single value of one of the many types. Manages memory for complex types, such as strings. Note: the destructor was left non-virtual so no virtual table will be created. This probably isn't necessary, but will avoid needless memory allocation. Since we'll never extend this class, not having a virtual destructor isn't a concern.

6.421.2 Member Enumeration Documentation

6.421.2.1 enum activemq::util::PrimitiveValueNode::PrimitiveType

Enumeration for the various primitive types.

Enumerator:

NULL_TYPE

BOOLEAN_TYPE
BYTE_TYPE
CHAR_TYPE
SHORT_TYPE
INTEGER_TYPE
LONG_TYPE
DOUBLE_TYPE
FLOAT_TYPE
STRING_TYPE
BYTE_ARRAY_TYPE
MAP_TYPE
LIST_TYPE
BIG_STRING_TYPE

6.421.3 Constructor & Destructor Documentation

6.421.3.1 activemq::util::PrimitiveValueNode::PrimitiveValueNode ()

Default Constructor, creates a value of the NULL_TYPE.

6.421.3.2 activemq::util::PrimitiveValueNode::PrimitiveValueNode (bool *value*)

Boolean Value Constructor.

Parameters:

value - the new value to store.

6.421.3.3 activemq::util::PrimitiveValueNode::PrimitiveValueNode (unsigned char *value*)

Byte Value Constructor.

Parameters:

value - the new value to store.

6.421.3.4 activemq::util::PrimitiveValueNode::PrimitiveValueNode (char *value*)

Char Value Constructor.

Parameters:

value - the new value to store.

6.421.3.5 activemq::util::PrimitiveValueNode::PrimitiveValueNode (short *value*)

Short Value Constructor.

Parameters:

value - the new value to store.

6.421.3.6 activemq::util::PrimitiveValueNode::PrimitiveValueNode (int *value*)

Int Value Constructor.

Parameters:

value - the new value to store.

6.421.3.7 activemq::util::PrimitiveValueNode::PrimitiveValueNode (long long *value*)

Long Value Constructor.

Parameters:

value - the new value to store.

6.421.3.8 activemq::util::PrimitiveValueNode::PrimitiveValueNode (float *value*)

Float Value Constructor.

Parameters:

value - the new value to store.

6.421.3.9 activemq::util::PrimitiveValueNode::PrimitiveValueNode (double *value*)

Double Value Constructor.

Parameters:

value - the new value to store.

6.421.3.10 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const char * *value*)

String Value Constructor.

Parameters:

value - the new value to store.

6.421.3.11 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::string & *value*)

String Value Constructor.

Parameters:

value - the new value to store.

6.421.3.12 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const std::vector< unsigned char > & *value*)

Byte Array Value Constructor.

Parameters:

value - the new value to store.

6.421.3.13 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::List< PrimitiveValueNode > & *value*)

Primtive List Constructor.

Parameters:

value - the new value to store.

6.421.3.14 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Primtive Map Value Constructor.

Parameters:

value - the new value to store.

6.421.3.15 activemq::util::PrimitiveValueNode::PrimitiveValueNode (const PrimitiveValueNode & *node*)

Copy constructor.

Parameters:

node The instance of another node to copy to this one.

6.421.3.16 activemq::util::PrimitiveValueNode::~PrimitiveValueNode () [inline]

6.421.4 Member Function Documentation

6.421.4.1 void activemq::util::PrimitiveValueNode::clear ()

Clears the value from this wrapper converting it back to a blank NULL_TYPE value.

6.421.4.2 bool activemq::util::PrimitiveValueNode::getBool () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Boolean value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.3 unsigned char activemq::util::PrimitiveValueNode::getByte () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Byte value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.4 std::vector<unsigned char> activemq::util::PrimitiveValueNode::getByteArray () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Byte Array value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.5 char activemq::util::PrimitiveValueNode::getChar () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Character value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.6 double activemq::util::PrimitiveValueNode::getDouble () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Double value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.7 float activemq::util::PrimitiveValueNode::getFloat () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Float value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.8 int activemq::util::PrimitiveValueNode::getInt () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Integer value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.9 const decaf::util::List<PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getList () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Primitive List value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.10 long long activemq::util::PrimitiveValueNode::getLong () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Long value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.11 const decaf::util::Map<std::string, PrimitiveValueNode>& activemq::util::PrimitiveValueNode::getMap () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Primitive Map value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.12 short activemq::util::PrimitiveValueNode::getShort () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the Short value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.13 std::string activemq::util::PrimitiveValueNode::getString () const throw (decaf::lang::exceptions::NoSuchElementException)

Gets the String value of this Node.

Returns:

value contained at the given index

Exceptions:

NoSuchElementException this node cannot be returned as the requested type.

6.421.4.14 PrimitiveType activemq::util::PrimitiveValueNode::getType () const [inline]

Gets the Value Type of this type wrapper.

Returns:

the PrimitiveType value for this wrapper.

6.421.4.15 PrimitiveValue activemq::util::PrimitiveValueNode::getValue () const [inline]

Gets the internal Primitive Value object from this wrapper.

Returns:

a copy of the contained **PrimitiveValue** (p. 2066)

6.421.4.16 PrimitiveValueNode& activemq::util::PrimitiveValueNode::operator= (const PrimitiveValueNode & *node*)

Assignment operator, copies the data from the other node.

Parameters:

node The instance of another node to copy to this one.

6.421.4.17 bool activemq::util::PrimitiveValueNode::operator== (const PrimitiveValueNode & *node*) const

Comparison Operator, compares this node to the other node.

Returns:

true if the values are the same false otherwise.

6.421.4.18 void activemq::util::PrimitiveValueNode::setBool (bool *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.19 void activemq::util::PrimitiveValueNode::setByte (unsigned char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.20 void activemq::util::PrimitiveValueNode::setByteArray (const std::vector< unsigned char > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.21 void activemq::util::PrimitiveValueNode::setChar (char *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.22 void activemq::util::PrimitiveValueNode::setDouble (double *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.23 void activemq::util::PrimitiveValueNode::setFloat (float *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.24 void activemq::util::PrimitiveValueNode::setInt (int *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.25 void activemq::util::PrimitiveValueNode::setList (const decaf::util::List< PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.26 void activemq::util::PrimitiveValueNode::setLong (long long *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.27 void activemq::util::PrimitiveValueNode::setMap (const decaf::util::Map< std::string, PrimitiveValueNode > & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.28 void activemq::util::PrimitiveValueNode::setShort (short *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.29 void activemq::util::PrimitiveValueNode::setString (const std::string & *value*)

Sets the value of this value node to the new value specified, this method overwrites any data that was previously at the index given.

Parameters:

value - the new value to assign to the element at index

6.421.4.30 void activemq::util::PrimitiveValueNode::setValue (const PrimitiveValue & *value*, PrimitiveType *valueType*)

Sets the internal PrimitiveVale object to the new value along with the tag for the type that it consists of.

Parameters:

value The value to set as the value contained in this Node.

valueType The type of the value being set into this one.

6.421.4.31 std::string activemq::util::PrimitiveValueNode::toString () const

Creates a string representation of this value.

Returns:

string value of this type wrapper.

The documentation for this class was generated from the following file:

- src/main/activemq/util/**PrimitiveValueNode.h**

6.422 decaf::security::Principal Class Reference

Base interface for a principal, which can represent an individual or organization.

`#include <src/main/decaf/security/Principal.h>`Inheritance diagram for decaf::security::Principal:

Public Member Functions

- virtual ~**Principal** ()
- virtual bool **equals** (const **Principal** &another) const =0

  Compares two principals to see if they are the same.

- virtual std::string **getName** () const =0

  Provides the name of this principal.

6.422.1 Detailed Description

Base interface for a principal, which can represent an individual or organization.

6.422.2 Constructor & Destructor Documentation

6.422.2.1 virtual decaf::security::Principal::~Principal () [inline, virtual]

6.422.3 Member Function Documentation

6.422.3.1 virtual bool decaf::security::Principal::equals (const Principal & *another*) const [pure virtual]

Compares two principals to see if they are the same.

Parameters:

another A principal to be tested for equality to this one.

Returns:

true if the given principal is equivalent to this one.

6.422.3.2 virtual std::string decaf::security::Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns:

the name of this principal.

Implemented in **decaf::security::auth::x500::X500Principal** (p. 2727).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**Principal.h**

6.423 activemq::commands::ProducerAck Class Reference

`#include <src/main/activemq/commands/ProducerAck.h>`Inheritance diagram for activemq::commands::ProducerAck:

Public Member Functions

- **ProducerAck** ()
- virtual ~**ProducerAck** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ProducerAck** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual int **getSize** () const
- virtual void **setSize** (int **size**)
- virtual bool **isProducerAck** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERACK** = 19

Protected Member Functions

- **ProducerAck** (const **ProducerAck** &)
- **ProducerAck** & **operator**= (const **ProducerAck** &)

Protected Attributes

- **Pointer< ProducerId > producerId**
- int **size**

6.423.1 Constructor & Destructor Documentation

6.423.1.1 activemq::commands::ProducerAck::ProducerAck (const ProducerAck &) [inline, protected]

6.423.1.2 activemq::commands::ProducerAck::ProducerAck ()

6.423.1.3 virtual activemq::commands::ProducerAck::~ProducerAck () [virtual]

6.423.2 Member Function Documentation

6.423.2.1 virtual ProducerAck* activemq::commands::ProducerAck::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.423.2.2 virtual void activemq::commands::ProducerAck::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.423.2.3 virtual bool activemq::commands::ProducerAck::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.423.2.4 virtual unsigned char activemq::commands::ProducerAck::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.423.2.5 virtual Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () [virtual]

6.423.2.6 virtual const Pointer<ProducerId>& activemq::commands::ProducerAck::getProducerId () const [virtual]

6.423.2.7 virtual int activemq::commands::ProducerAck::getSize () const [virtual]

6.423.2.8 virtual bool activemq::commands::ProducerAck::isProducerAck () const [inline, virtual]

Returns:

an answer of true to the **isProducerAck()** (p. 2088) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.423.2.9 ProducerAck& activemq::commands::ProducerAck::operator= (const ProducerAck &) [inline, protected]

6.423.2.10 virtual void activemq::commands::ProducerAck::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]

6.423.2.11 virtual void activemq::commands::ProducerAck::setSize (int *size*) [virtual]

6.423.2.12 virtual std::string activemq::commands::ProducerAck::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.423.2.13 virtual Pointer<Command> activemq::commands::ProducerAck::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.423.3 Field Documentation

6.423.3.1 const unsigned char activemq::commands::ProducerAck::ID_PRODUCERACK = 19 [static]

6.423.3.2 Pointer<ProducerId> activemq::commands::ProducerAck::producerId [protected]

6.423.3.3 int activemq::commands::ProducerAck::size [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerAck.h**

6.424 activemq::wireformat::openwire::marshal::v3::ProducerAckMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2090).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual ~**ProducerAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.424.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2090). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.424.2 Constructor & Destructor Documentation

6.424.2.1 activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::ProducerAckMarsh () [inline]

6.424.2.2 virtual activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::~ProducerAckMars () [inline, virtual]

6.424.3 Member Function Documentation

6.424.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.424.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.424.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.424.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.424.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.424.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.424.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerAckMarshaller.h**

6.425 activemq::wireformat::openwire::marshal::v1::ProducerAckMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2094).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual ~**ProducerAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.425.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2094). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.425.2 Constructor & Destructor Documentation

6.425.2.1 activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::ProducerAckMarsh () [inline]

6.425.2.2 virtual activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::~ProducerAckMars () [inline, virtual]

6.425.3 Member Function Documentation

6.425.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.425.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.425.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.425.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.425.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.425.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.425.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerAckMarshaller.h**

6.426 activemq::wireformat::openwire::marshal::v2::ProducerAckMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2098).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerAckMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller:

Public Member Functions

- **ProducerAckMarshaller** ()
- virtual ~**ProducerAckMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.426.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2098). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.426.2 Constructor & Destructor Documentation

6.426.2.1 activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::ProducerAckMarsh () [inline]

6.426.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::~ProducerAckMars () [inline, virtual]

6.426.3 Member Function Documentation

6.426.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.426.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.426.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.426.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.426.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.426.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.426.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerAckMarshaller.h**

6.427 activemq::cmsutil::ProducerCallback Class Reference

Callback for sending a message to a CMS destination.

`#include <src/main/activemq/cmsutil/ProducerCallback.h>`Inheritance diagram for activemq::cmsutil::ProducerCallback:

Public Member Functions

- virtual ~**ProducerCallback** ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer)=0 throw (cms::CMSException)

  Execute an action given a session and producer.

6.427.1 Detailed Description

Callback for sending a message to a CMS destination.

6.427.2 Constructor & Destructor Documentation

6.427.2.1 virtual activemq::cmsutil::ProducerCallback::~ProducerCallback () [inline, virtual]

6.427.3 Member Function Documentation

6.427.3.1 virtual void activemq::cmsutil::ProducerCallback::doInCms (cms::Session * *session*, cms::MessageProducer * *producer*) throw (cms::CMSException) [pure virtual]

Execute an action given a session and producer.

Parameters:

session the CMS `Session`

producer the CMS `Producer`

Exceptions:

cms::CMSException *(*p. ***850)*** if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::SendExecutor** (p. 2267).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ProducerCallback.h**

6.428 activemq::cmsutil::CmsTemplate::ProducerExecutor Class Reference

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate::ProducerExecutor:

Public Member Functions

- **ProducerExecutor** (**ProducerCallback** *__action__, **CmsTemplate** *__parent__, **cms::Destination** *__destination__)
- virtual ~**ProducerExecutor** ()
- virtual void **doInCms** (**cms::Session** *session) throw (cms::CMSException)

  Execute any number of operations against the supplied CMS session.

- virtual **cms::Destination** * **getDestination** (**cms::Session** *session AMQCPP_UNUSED) throw (cms::CMSException)

Protected Attributes

- **ProducerCallback** * **action**
- **CmsTemplate** * **parent**
- **cms::Destination** * **destination**

6.428.1 Constructor & Destructor Documentation

6.428.1.1 activemq::cmsutil::CmsTemplate::ProducerExecutor::ProducerExecutor (ProducerCallback * *action*, CmsTemplate * *parent*, cms::Destination * *destination*) `[inline]`

6.428.1.2 virtual activemq::cmsutil::CmsTemplate::ProducerExecutor::~ProducerExecutor () `[inline, virtual]`

6.428.2 Member Function Documentation

6.428.2.1 virtual void activemq::cmsutil::CmsTemplate::ProducerExecutor::doInCms (cms::Session * *session*) throw (cms::CMSException) `[virtual]`

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS `Session`

Exceptions:

cms::CMSException *(*p. ***850)*** if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 2283).

6.428.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

6.428.3 Field Documentation

6.428.3.1 ProducerCallback* activemq::cmsutil::CmsTemplate::ProducerExecutor::action [protected]

6.428.3.2 cms::Destination* activemq::cmsutil::CmsTemplate::ProducerExecutor::destination [protected]

6.428.3.3 CmsTemplate* activemq::cmsutil::CmsTemplate::ProducerExecutor::parent [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.429 activemq::commands::ProducerId Class Reference

`#include <src/main/activemq/commands/ProducerId.h>`Inheritance diagram for activemq::commands::ProducerId:

Public Types

- typedef **decaf::lang::PointerComparator< ProducerId > COMPARATOR**

Public Member Functions

- **ProducerId** ()
- **ProducerId** (const **ProducerId** &other)
- **ProducerId** (const **SessionId** &**sessionId**, long long consumerId)
- virtual ~**ProducerId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ProducerId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- const **Pointer< SessionId >** & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &**connectionId**)
- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual long long **getSessionId** () const
- virtual void **setSessionId** (long long **sessionId**)
- virtual int **compareTo** (const **ProducerId** &**value**) const
- virtual bool **equals** (const **ProducerId** &**value**) const
- virtual bool **operator==** (const **ProducerId** &**value**) const
- virtual bool **operator<** (const **ProducerId** &**value**) const
- **ProducerId** & **operator=** (const **ProducerId** &other)

Static Public Attributes

- static const unsigned char **ID_PRODUCERID** = 123

Protected Attributes

- std::string **connectionId**
- long long **value**
- long long **sessionId**

6.429.1 Member Typedef Documentation

6.429.1.1 typedef decaf::lang::PointerComparator<ProducerId> activemq::commands::ProducerId::COMPARATOR

6.429.2 Constructor & Destructor Documentation

6.429.2.1 activemq::commands::ProducerId::ProducerId ()

6.429.2.2 activemq::commands::ProducerId::ProducerId (const ProducerId & *other*)

6.429.2.3 activemq::commands::ProducerId::ProducerId (const SessionId & *sessionId*, long long *consumerId*) [inline]

References activemq::commands::SessionId::getConnectionId(), and activemq::commands::SessionId::getValue().

6.429.2.4 virtual activemq::commands::ProducerId::~ProducerId () [virtual]

6.429.3 Member Function Documentation

6.429.3.1 virtual ProducerId* activemq::commands::ProducerId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.429.3.2 virtual int activemq::commands::ProducerId::compareTo (const ProducerId & *value*) const [virtual]

6.429.3.3 virtual void activemq::commands::ProducerId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.429.3.4 virtual bool activemq::commands::ProducerId::equals (const ProducerId & *value*) const [virtual]

6.429.3.5 virtual bool activemq::commands::ProducerId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.429.3.6 virtual std::string& activemq::commands::ProducerId::getConnectionId () [virtual]

6.429.3.7 virtual const std::string& activemq::commands::ProducerId::getConnectionId () const [virtual]

6.429.3.8 virtual unsigned char activemq::commands::ProducerId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.429.3.9 const Pointer<SessionId>& activemq::commands::ProducerId::getParentId () const

6.429.3.10 virtual long long activemq::commands::ProducerId::getSessionId () const [virtual]

6.429.3.11 virtual long long activemq::commands::ProducerId::getValue () const [virtual]

6.429.3.12 virtual bool activemq::commands::ProducerId::operator< (const ProducerId & *value*) const [virtual]

6.429.3.13 ProducerId& activemq::commands::ProducerId::operator= (const ProducerId & *other*)

6.429.3.14 virtual bool activemq::commands::ProducerId::operator== (const ProducerId & *value*) const [virtual]

6.429.3.15 virtual void activemq::commands::ProducerId::setConnectionId (const std::string & *connectionId*) [virtual]

6.429.3.16 virtual void activemq::commands::ProducerId::setSessionId (long long *sessionId*) [virtual]

6.429.3.17 virtual void activemq::commands::ProducerId::setValue (long long *value*) [virtual]

6.429.3.18 virtual std::string activemq::commands::ProducerId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.429.4 Field Documentation

6.429.4.1 std::string activemq::commands::ProducerId::connectionId [protected]

6.429.4.2 const unsigned char activemq::commands::ProducerId::ID_PRODUCERID = 123 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.429.4.3 long long activemq::commands::ProducerId::sessionId [protected]

6.429.4.4 long long activemq::commands::ProducerId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerId.h**

6.430 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshall Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2110).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual ~**ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.430.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2110). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.430.2 Constructor & Destructor Documentation

6.430.2.1 activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::ProducerIdMarshalle () [inline]

6.430.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::~ProducerIdMarshal () [inline, virtual]

6.430.3 Member Function Documentation

6.430.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.430.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.430.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.430.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.430.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.430.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.430.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerIdMarshaller.h**

6.431 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshall Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2114).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual ~**ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.431.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2114). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.431.2 Constructor & Destructor Documentation

6.431.2.1 activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::ProducerIdMarshalle () [inline]

6.431.2.2 virtual activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::~ProducerIdMarshal () [inline, virtual]

6.431.3 Member Function Documentation

6.431.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.431.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.431.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.431.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.431.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.431.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.431.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerIdMarshaller.h**

6.432 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshall Class Reference

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2118).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller:

Public Member Functions

- **ProducerIdMarshaller** ()
- virtual ~**ProducerIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.432.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerIdMarshaller** (p. 2118). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.432.2 Constructor & Destructor Documentation

6.432.2.1 activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::ProducerIdMarshalle () [inline]

6.432.2.2 virtual activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::~ProducerIdMarshal () [inline, virtual]

6.432.3 Member Function Documentation

6.432.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.432.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.432.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.432.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.432.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.432.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.432.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerIdMarshaller.h**

6.433 activemq::commands::ProducerInfo Class Reference

`#include <src/main/activemq/commands/ProducerInfo.h>`Inheritance diagram for activemq::commands::ProducerInfo:

Public Member Functions

- **ProducerInfo** ()
- virtual ~**ProducerInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ProducerInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ProducerId** > & **getProducerId** () const
- virtual **Pointer**< **ProducerId** > & **getProducerId** ()
- virtual void **setProducerId** (const **Pointer**< **ProducerId** > &**producerId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** () const
- virtual std::vector< **decaf::lang::Pointer**< **BrokerId** > > & **getBrokerPath** ()
- virtual void **setBrokerPath** (const std::vector< **decaf::lang::Pointer**< **BrokerId** > > &**brokerPath**)
- virtual bool **isDispatchAsync** () const
- virtual void **setDispatchAsync** (bool **dispatchAsync**)
- virtual int **getWindowSize** () const
- virtual void **setWindowSize** (int **windowSize**)
- virtual bool **isProducerInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_PRODUCERINFO** = 6

Protected Member Functions

- **ProducerInfo** (const **ProducerInfo** &)
- **ProducerInfo** & **operator**= (const **ProducerInfo** &)

Protected Attributes

- **Pointer< ProducerId > producerId**
- **Pointer< ActiveMQDestination > destination**
- std::vector< **decaf::lang::Pointer< BrokerId > > brokerPath**
- bool **dispatchAsync**
- int **windowSize**

6.433.1 Constructor & Destructor Documentation

6.433.1.1 activemq::commands::ProducerInfo::ProducerInfo (const ProducerInfo &) [inline, protected]

6.433.1.2 activemq::commands::ProducerInfo::ProducerInfo ()

6.433.1.3 virtual activemq::commands::ProducerInfo::~ProducerInfo () [virtual]

6.433.2 Member Function Documentation

6.433.2.1 virtual ProducerInfo* activemq::commands::ProducerInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.433.2.2 virtual void activemq::commands::ProducerInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.433.2.3 virtual bool activemq::commands::ProducerInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.433.2.4 virtual std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () [virtual]

6.433.2.5 virtual const std::vector< decaf::lang::Pointer<BrokerId> >& activemq::commands::ProducerInfo::getBrokerPath () const [virtual]

6.433.2.6 virtual unsigned char activemq::commands::ProducerInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.433.2.7 virtual Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () [virtual]

6.433.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::ProducerInfo::getDestination () const [virtual]

6.433.2.9 virtual Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () [virtual]

6.433.2.10 virtual const Pointer<ProducerId>& activemq::commands::ProducerInfo::getProducerId () const [virtual]

6.433.2.11 virtual int activemq::commands::ProducerInfo::getWindowSize () const [virtual]

6.433.2.12 virtual bool activemq::commands::ProducerInfo::isDispatchAsync () const [virtual]

6.433.2.13 virtual bool activemq::commands::ProducerInfo::isProducerInfo () const [inline, virtual]

Returns:

an answer of true to the **isProducerInfo()** (p. 2125) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.433.2.14 ProducerInfo& activemq::commands::ProducerInfo::operator= (const ProducerInfo &) [inline, protected]

6.433.2.15 virtual void activemq::commands::ProducerInfo::setBrokerPath (const std::vector< decaf::lang::Pointer< BrokerId > > & *brokerPath*) [virtual]

6.433.2.16 virtual void activemq::commands::ProducerInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.433.2.17 virtual void activemq::commands::ProducerInfo::setDispatchAsync (bool *dispatchAsync*) [virtual]

6.433.2.18 virtual void activemq::commands::ProducerInfo::setProducerId (const Pointer< ProducerId > & *producerId*) [virtual]

6.433.2.19 virtual void activemq::commands::ProducerInfo::setWindowSize (int *windowSize*) [virtual]

6.433.2.20 virtual std::string activemq::commands::ProducerInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.433.2.21 virtual Pointer<Command> activemq::commands::ProducerInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.433.3 Field Documentation

6.433.3.1 std::vector< decaf::lang::Pointer<BrokerId> > activemq::commands::ProducerInfo::brokerPath [protected]

6.433.3.2 Pointer<ActiveMQDestination> activemq::commands::ProducerInfo::destination [protected]

6.433.3.3 bool activemq::commands::ProducerInfo::dispatchAsync [protected]

6.433.3.4 const unsigned char activemq::commands::ProducerInfo::ID_PRODUCERINFO = 6 [static]

6.433.3.5 Pointer<ProducerId> activemq::commands::ProducerInfo::producerId [protected]

6.433.3.6 int activemq::commands::ProducerInfo::windowSize [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ProducerInfo.h**

6.434 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2127).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ProducerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.434.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2127). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.434.2 Constructor & Destructor Documentation

6.434.2.1 activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::ProducerInfoMarsh () [inline]

6.434.2.2 virtual activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::~ProducerInfoMar () [inline, virtual]

6.434.3 Member Function Documentation

6.434.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.434.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.434.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.434.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.434.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.434.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.434.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ProducerInfoMarshaller.h**

6.435 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2131).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ProducerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.435.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2131). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.435.2 Constructor & Destructor Documentation

6.435.2.1 activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::ProducerInfoMarsh () [inline]

6.435.2.2 virtual activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::~ProducerInfoMar () [inline, virtual]

6.435.3 Member Function Documentation

6.435.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.435.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.435.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.435.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.435.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.435.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.435.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ProducerInfoMarshaller.h**

6.436 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarsha Class Reference

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2135).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ProducerInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller:

Public Member Functions

- **ProducerInfoMarshaller** ()
- virtual ~**ProducerInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.436.1 Detailed Description

Marshaling code for Open Wire Format for **ProducerInfoMarshaller** (p. 2135). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.436.2 Constructor & Destructor Documentation

6.436.2.1 activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::ProducerInfoMarsh () [inline]

6.436.2.2 virtual activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::~ProducerInfoMar () [inline, virtual]

6.436.3 Member Function Documentation

6.436.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.436.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.436.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.436.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.436.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.436.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.436.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ProducerInfoMarshaller.h**

6.437 activemq::state::ProducerState Class Reference

`#include <src/main/activemq/state/ProducerState.h>`

Public Member Functions

- **ProducerState** (const **Pointer**< **ProducerInfo** > &info)
- virtual ~**ProducerState** ()
- std::string **toString** () const
- const **Pointer**< **ProducerInfo** > & **getInfo** () const

6.437.1 Constructor & Destructor Documentation

6.437.1.1 activemq::state::ProducerState::ProducerState (const Pointer< ProducerInfo > & *info*)

6.437.1.2 virtual activemq::state::ProducerState::~ProducerState () [virtual]

6.437.2 Member Function Documentation

6.437.2.1 const Pointer<ProducerInfo>& activemq::state::ProducerState::getInfo () const [inline]

6.437.2.2 std::string activemq::state::ProducerState::toString () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**ProducerState.h**

6.438 decaf::util::Properties Class Reference

Java-like properties class for mapping string names to string values.

`#include <src/main/decaf/util/Properties.h>`

Public Member Functions

- **Properties** ()
- **Properties** (const **Properties** &src)
- virtual ~**Properties** ()
- **Properties** & **operator**= (const **Properties** &src)

  Assignment Operator.

- bool **isEmpty** () const

  Returns true if the properties object is empty.

- std::size_t **size** () const
- const char * **getProperty** (const std::string &name) const

  Looks up the value for the given property.

- std::string **getProperty** (const std::string &name, const std::string &defaultValue) const

  Looks up the value for the given property.

- void **setProperty** (const std::string &name, const std::string &value)

  Sets the value for a given property.

- bool **hasProperty** (const std::string &name) const

  Check to see if the Property exists in the set.

- void **remove** (const std::string &name)

  Removes the property with the given name.

- std::vector< std::pair< std::string, std::string > > **toArray** () const

  Method that serializes the contents of the property map to an array.

- void **copy** (const **Properties** &source)

  *Copies the contents of the given properties object to this one, if the given **Properties** (p. 2140) instance in NULL then this **List** (p. 1591) is not modified.*

- **Properties** * **clone** () const

  Clones this object.

- void **clear** ()

  Clears all properties from the map.

- bool **equals** (const **Properties** &source) const

  *Test whether two **Properties** (p. 2140) objects are equivalent.*

- std::string **toString** () const

*Formats the contents of the **Properties** (p. 2140) Object into a string that can be logged, etc.*

- void **load** (**decaf::io::InputStream** *stream) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

  Reads a property list (key and element pairs) from the input byte stream.

- void **load** (**decaf::io::Reader** *reader) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

  Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format.

- void **store** (**decaf::io::OutputStream** *out, const std::string &comment) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

  *Writes this property list (key and element pairs) in this **Properties** (p. 2140) table to the output stream in a format suitable for loading into a **Properties** (p. 2140) table using the load(InputStream) method.*

- void **store** (**decaf::io::Writer** *writer, const std::string &comments) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

  *Writes this property list (key and element pairs) in this **Properties** (p. 2140) table to the output character stream in a format that can be read by the load(Reader) method.*

Protected Attributes

- std::auto_ptr< **Properties** > **defaults**

  Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

6.438.1 Detailed Description

Java-like properties class for mapping string names to string values. The **Properties** (p. 2140) list contains a key value pair of properties that can be loaded and stored to a stream. Each **Properties** (p. 2140) instance can contain an **internal** (p. 95) **Properties** (p. 2140) list that contains default values for keys not found in the **Properties** (p. 2140) **List** (p. 1591).

The **Properties** (p. 2140) list if a Thread Safe class, it can be shared amongst objects in multiple threads without the need for additional synchronization.

Since:

1.0

6.438.2 Constructor & Destructor Documentation

6.438.2.1 decaf::util::Properties::Properties ()

6.438.2.2 decaf::util::Properties::Properties (const Properties & *src*)

6.438.2.3 virtual decaf::util::Properties::~Properties () [virtual]

6.438.3 Member Function Documentation

6.438.3.1 void decaf::util::Properties::clear ()

Clears all properties from the map.

6.438.3.2 Properties* decaf::util::Properties::clone () const

Clones this object.

Returns:

a replica of this object.

6.438.3.3 void decaf::util::Properties::copy (const Properties & *source*)

Copies the contents of the given properties object to this one, if the given **Properties** (p. 2140) instance in NULL then this **List** (p. 1591) is not modified.

Parameters:

source The source properties object.

6.438.3.4 bool decaf::util::Properties::equals (const Properties & *source*) const

Test whether two **Properties** (p. 2140) objects are equivalent. Two **Properties** (p. 2140) Objects are considered equivalent when they each contain the same number of elements and each key / value pair contained within the two are equal.

This comparison does not check the contents of the Defaults instance.

Parameters:

source The **Properties** (p. 2140) object to compare this instance to.

Returns:

true if the contents of the two **Properties** (p. 2140) objects are the same.

6.438.3.5 std::string decaf::util::Properties::getProperty (const std::string & *name*, const std::string & *defaultValue*) const

Looks up the value for the given property.

Parameters:

name the name of the property to be looked up.

defaultValue The value to be returned if the given property does not exist.

Returns:

The value of the property specified by `name`, if it exists, otherwise the `defaultValue`.

6.438.3.6 const char* decaf::util::Properties::getProperty (const std::string & *name*) const

Looks up the value for the given property.

Parameters:

name The name of the property to be looked up.

Returns:

the value of the property with the given name, if it exists. If it does not exist, returns NULL.

Referenced by decaf::util::logging::LogManager::getProperty().

6.438.3.7 bool decaf::util::Properties::hasProperty (const std::string & *name*) const

Check to see if the Property exists in the set.

Parameters:

name - property name to check for in this properties set.

Returns:

true if property exists, false otherwise.

6.438.3.8 bool decaf::util::Properties::isEmpty () const

Returns true if the properties object is empty.

Returns:

true if empty

6.438.3.9 void decaf::util::Properties::load (decaf::io::Reader * *reader*) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input character stream in a simple line-oriented format. **Properties** (p. 2140) are processed in terms of lines. There are two kinds of line, natural lines and logical lines. A natural line is defined as a line of characters that is terminated either by a set of line terminator characters (

or or

) or by the end of the stream. A natural line may be either a blank line, a comment line, or hold all or some of a key-element pair. A logical line holds all the data of a key-element pair, which may be spread out across several adjacent natural lines by escaping the line terminator sequence with a backslash character \. Note that a comment line cannot be extended in this manner; every natural line that is a comment must have its own comment indicator, as described below. Lines are read from input until the end of the stream is reached.

A natural line that contains only white space characters is considered blank and is ignored. A comment line has an ASCII '#' or '!' as its first non-white space character; comment lines are also ignored and do not encode key-element information. In addition to line terminators, this format considers the characters space (' '), tab ('), and form feed (') to be white space.

If a logical line is spread across several natural lines, the backslash escaping the line terminator sequence, the line terminator sequence, and any white space at the start of the following line have no affect on the key or element values. The remainder of the discussion of key and element parsing (when loading) will assume all the characters constituting the key and element appear on a single natural line after line continuation characters have been removed. Note that it is not sufficient to only examine the character preceding a line terminator sequence to decide if the line terminator is escaped; there must be an odd number of contiguous backslashes for the line terminator to be escaped. Since the input is processed from left to right, a non-zero even number of 2n contiguous backslashes before a line terminator (or elsewhere) encodes n backslashes after escape processing.

The key contains all of the characters in the line starting with the first non-white space character and up to, but not including, the first unescaped '=', ':', or white space character other than a line terminator. All of these key termination characters may be included in the key by escaping them with a preceding backslash character; for example,

\:\=

would be the two-character key ":=". Line terminator characters can be included using and

escape sequences. Any white space after the key is skipped; if the first non-white space character after the key is '=' or ':', then it is ignored and any white space characters after it are also skipped. All remaining characters on the line become part of the associated element string; if there are no remaining characters, the element is the empty string "". Once the raw character sequences constituting the key and element are identified, escape processing is performed as described above.

As an example, each of the following three lines specifies the key "Truth" and the associated element value "Beauty":

Truth = Beauty Truth:Beauty Truth :Beauty

As another example, the following three lines specify a single property:

fruits apple, banana, pear, \ cantaloupe, watermelon, \ kiwi, mango

The key is "fruits" and the associated element is: "apple, banana, pear, cantaloupe, watermelon, kiwi, mango"

Note that a space appears before each \ so that a space will appear after each comma in the final result; the \, line terminator, and leading white space on the continuation line are merely discarded and are not replaced by one or more other characters.

As a third example, the line:

cheeses

specifies that the key is "cheeses" and the associated element is the empty string "".

Characters in keys and elements can be represented in escape sequences similar to those used

for character and string literals (see §3.3 and §3.10.6 of the Java Language Specification). The differences from the character escape sequences and Unicode escapes used for characters and strings are:

- Octal escapes are not recognized.
- The character sequence **does** not represent a backspace character.
- The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a C++ string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.
- Escapes are not necessary for single and double quotes; however, by the rule above, single and double quote characters preceded by a backslash still yield single and double quote characters, respectively.

This method does not close the Reader upon its return.

Parameters:

reader The Reader that provides an character stream as input.

Exceptions:

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.438.3.10 void decaf::util::Properties::load (decaf::io::InputStream * *stream*) throw (decaf::io::IOException, decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Reads a property list (key and element pairs) from the input byte stream. The input stream is in a simple line-oriented format as specified in load(Reader) and is assumed to use the ISO 8859-1 character encoding.

This method does not close the stream upon its return.

Parameters:

stream The stream to read the properties data from.

Exceptions:

IOException if there is an error while reading from the stream.

IllegalArgumentException if malformed data is found while reading the properties.

NullPointerException if the passed stream is Null.

6.438.3.11 Properties& decaf::util::Properties::operator= (const Properties & *src*)

Assignment Operator.

Parameters:

src The **Properties** (p. 2140) list to copy to this **List** (p. 1591).

Returns:

a reference to this **List** (p. 1591) for use in chaining.

6.438.3.12 void decaf::util::Properties::remove (const std::string & *name*)

Removes the property with the given name.

Parameters:

name the name of the property to remove.

6.438.3.13 void decaf::util::Properties::setProperty (const std::string & *name*, const std::string & *value*)

Sets the value for a given property. If the property already exists, overwrites the value.

Parameters:

name The name of the value to be written.

value The value to be written.

6.438.3.14 std::size_t decaf::util::Properties::size () const

Returns:

The number of **Properties** (p. 2140) in this **Properties** (p. 2140) Object.

6.438.3.15 void decaf::util::Properties::store (decaf::io::Writer * *writer*, const std::string & *comments*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

Writes this property list (key and element pairs) in this **Properties** (p. 2140) table to the output character stream in a format that can be read by the load(Reader) method. **Properties** (p. 2140) from the defaults table of this **Properties** (p. 2140) table (if any) are not written out by this method.

If the comments argument is not empty, then an ASCII # character, the comments string, and a line separator are first written to the output stream. Thus, the comments can serve as an identifying comment. Any one of a line feed ('

'), a carriage return (''), or a carriage return followed immediately by a line feed in comments is replaced by a line separator generated by the Writer and if the next character in comments is not character # or character ! then an ASCII # is written out after that line separator.

Next, a comment line is always written, consisting of an ASCII # character, the current date and time (as if produced by the toString method of **Date** (p. 1179) for the current time), and a line separator as generated by the Writer.

Then every entry in this **Properties** (p. 2140) table is written out, one per line. For each entry the key string is written, then an ASCII =, then the associated element string. For the key, all space characters are written with a preceding \ character. For the element, leading space characters, but not embedded or trailing space characters, are written with a preceding \ character. The key and element characters #, !, =, and : are written with a preceding backslash to ensure that they are properly loaded.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters:

writer The Writer instance to use to output the properties.

comments A description of these properties that is written before writing the properties.

Exceptions:

IOException if there is an error while writing from the stream.

NullPointerException if the passed stream is Null.

6.438.3.16 void decaf::util::Properties::store (decaf::io::OutputStream * *out*, const std::string & *comment*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

Writes this property list (key and element pairs) in this **Properties** (p. 2140) table to the output stream in a format suitable for loading into a **Properties** (p. 2140) table using the load(InputStream) method. **Properties** (p. 2140) from the defaults table of this **Properties** (p. 2140) table (if any) are not written out by this method.

This method outputs the comments, properties keys and values in the same format as specified in store(Writer), with the following differences:

- The stream is written using the ISO 8859-1 character encoding.
- Characters not in Latin-1 in the comments are written as for their appropriate unicode hexadecimal value xxxx.
- Characters less than and characters greater than in property keys or values are written as for the appropriate hexadecimal value xxxx.

After the entries have been written, the output stream is flushed. The output stream remains open after this method returns.

Parameters:

out The OutputStream instance to write the properties to.

comment A description of these properties that is written to the output stream.

Exceptions:

IOException if there is an error while writing from the stream.

NullPointerException if the passed stream is Null.

6.438.3.17 std::vector< std::pair< std::string, std::string > > decaf::util::Properties::toArray () const

Method that serializes the contents of the property map to an array.

Returns:

list of pairs where the first is the name and the second is the value.

6.438.3.18 std::string decaf::util::Properties::toString () const

Formats the contents of the **Properties** (p. 2140) Object into a string that can be logged, etc.

Returns:

string value of this object.

6.438.4 Field Documentation

6.438.4.1 std::auto_ptr<Properties> decaf::util::Properties::defaults [protected]

Default list used to answer for any keys not found in the properties list, can be filled in by another implementation of this class.

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Properties.h**

6.439 decaf::util::logging::PropertiesChangeListener Class Reference

Defines the interface that classes can use to listen for change events on **Properties** (p. 2140).

`#include <src/main/decaf/util/logging/PropertiesChangeListener.h>`

Public Member Functions

- virtual ~**PropertiesChangeListener** ()
- virtual void **onPropertyChanged** (const std::string &name, const std::string &oldValue, const std::string &newValue)=0

  Change Event, called when a property is changed.

6.439.1 Detailed Description

Defines the interface that classes can use to listen for change events on **Properties** (p. 2140).

6.439.2 Constructor & Destructor Documentation

6.439.2.1 virtual decaf::util::logging::PropertiesChangeListener::~PropertiesChangeListener () [inline, virtual]

6.439.3 Member Function Documentation

6.439.3.1 virtual void decaf::util::logging::PropertiesChangeListener::onPropertyChanged (const std::string & *name*, const std::string & *oldValue*, const std::string & *newValue*) [pure virtual]

Change Event, called when a property is changed.

Parameters:

name - Name of the Property

oldValue - Old Value of the Property

newValue - New Value of the Property

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**PropertiesChangeListener.h**

6.440 decaf::net::ProtocolException Class Reference

`#include <src/main/decaf/net/ProtocolException.h>`Inheritance diagram for decaf::net::ProtocolException:

Public Member Functions

- **ProtocolException** () throw ()
  Default Constructor.
- **ProtocolException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **ProtocolException** (const **ProtocolException** &ex) throw ()
  Copy Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **ProtocolException** (const std::exception ***cause**) throw ()
  Constructor.
- **ProtocolException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **ProtocolException** * **clone** () const
  Clones this exception.
- virtual ~**ProtocolException** () throw ()

6.440.1 Constructor & Destructor Documentation

6.440.1.1 decaf::net::ProtocolException::ProtocolException () throw () [inline]

Default Constructor.

6.440.1.2 decaf::net::ProtocolException::ProtocolException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.440.1.3 decaf::net::ProtocolException::ProtocolException (const ProtocolException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.440.1.4 decaf::net::ProtocolException::ProtocolException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.440.1.5 decaf::net::ProtocolException::ProtocolException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.440.1.6 decaf::net::ProtocolException::ProtocolException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.440.1.7 virtual decaf::net::ProtocolException::~ProtocolException () throw () [inline, virtual]

6.440.2 Member Function Documentation

6.440.2.1 virtual ProtocolException* decaf::net::ProtocolException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ProtocolException.h**

6.441 decaf::security::PublicKey Class Reference

A public key.

`#include <src/main/decaf/security/PublicKey.h>`Inheritance diagram for decaf::security::PublicKey:

Public Member Functions

- virtual ~**PublicKey** ()

6.441.1 Detailed Description

A public key. This interface contains no methods or constants. It merely serves to group (and provide type safety for) all public key interfaces.

6.441.2 Constructor & Destructor Documentation

6.441.2.1 virtual decaf::security::PublicKey::~PublicKey () `[inline, virtual]`

The documentation for this class was generated from the following file:

- src/main/decaf/security/**PublicKey.h**

6.442 decaf::util::Queue< E > Class Template Reference

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

`#include <src/main/decaf/util/Queue.h>`Inheritance diagram for decaf::util::Queue< E >:

Public Member Functions

- virtual ~**Queue** ()
- virtual const E & **getEmptyMarker** () const =0

  *Returns a reference to the Marker value that is returned from methods that do not throw an exception when there is no element in the **Queue** (p. 2154) to return.*

- virtual bool **offer** (const E &value)=0

  Inserts the specified element into the queue provided that the condition allows such an operation.

- virtual E **poll** ()=0

  Gets and removes the element in the head of the queue, or returns null if there is no element in the queue.

- virtual E **remove** ()=0 throw (decaf::lang::exceptions::NoSuchElementException)

  Gets and removes the element in the head of the queue.

- virtual const E & **peek** () const =0

  Gets but not removes the element in the head of the queue.

- virtual const E & **element** () const =0 throw (decaf::lang::exceptions::NoSuchElementException)

  Gets but not removes the element in the head of the queue.

6.442.1 Detailed Description

template<typename E> class decaf::util::Queue< E >

A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection. Generally, a queue orders its elements by means of first-in-first-out. While priority queue orders its elements according to a comparator specified or the elements' natural order. Furthermore, a stack orders its elements last-in-first out.

Queue (p. 2154) does not provide blocking queue methods, which will block until the operation of the method is allowed. BlockingQueue interface defines such methods.

Certain methods in the **Queue** (p. 2154) interface return a special value instead of throwing an exception if there is no element in the **Queue** (p. 2154) to return, this special value can be obtained by calling the **Queue** (p. 2154) method `getEmptyMarker`.

6.442.2 Constructor & Destructor Documentation

6.442.2.1 template<typename E > virtual decaf::util::Queue< E >::~Queue () [inline, virtual]

6.442.3 Member Function Documentation

6.442.3.1 template<typename E > virtual const E& decaf::util::Queue< E >::element () const throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Gets but not removes the element in the head of the queue. Throws a NoSuchElementException if there is no element in the queue.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if there is no element in the queue.

Implemented in **decaf::util::AbstractQueue< E >** (p. 138).

6.442.3.2 template<typename E > virtual const E& decaf::util::Queue< E >::getEmptyMarker () const [pure virtual]

Returns a reference to the Marker value that is returned from methods that do not throw an exception when there is no element in the **Queue** (p. 2154) to return. The empty marker is usually an instance of the contained element initialized using the default constructor (if its a Class) or the default value for the primitive type.

Returns:

a value that indicates that the **Queue** (p. 2154) is empty.

Referenced by decaf::util::AbstractQueue< E >::clear(), decaf::util::AbstractQueue< E >::element(), and decaf::util::AbstractQueue< E >::remove().

6.442.3.3 template<typename E > virtual bool decaf::util::Queue< E >::offer (const E & *value*) [pure virtual]

Inserts the specified element into the queue provided that the condition allows such an operation. The method is generally preferable to the collection.add(E), since the latter might throw an exception if the operation fails.

Parameters:

value the specified element to insert into the queue.

Returns:

true if the operation succeeds and false if it fails.

Referenced by decaf::util::AbstractQueue< E >::add().

6.442.3.4 template<typename E > virtual const E& decaf::util::Queue< E >::peek () const [pure virtual]

Gets but not removes the element in the head of the queue.

Returns:

the element in the head of the queue or null if there is no element in the queue.

Referenced by decaf::util::AbstractQueue< E >::element().

6.442.3.5 template<typename E > virtual E decaf::util::Queue< E >::poll () [pure virtual]

Gets and removes the element in the head of the queue, or returns null if there is no element in the queue.

Returns:

the element in the head of the queue or null if there is no element in the queue.

Referenced by decaf::util::AbstractQueue< E >::clear(), and decaf::util::AbstractQueue< E >::remove().

6.442.3.6 template<typename E > virtual E decaf::util::Queue< E >::remove () throw (decaf::lang::exceptions::NoSuchElementException) [pure virtual]

Gets and removes the element in the head of the queue. Throws a NoSuchElementException if there is no element in the queue.

Returns:

the element in the head of the queue.

Exceptions:

NoSuchElementException if there is no element in the queue.

Implemented in **decaf::util::AbstractQueue< E >** (p. 138).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Queue.h**

6.443 cms::Queue Class Reference

An interface encapsulating a provider-specific queue name.

`#include <src/main/cms/Queue.h>`Inheritance diagram for cms::Queue:

Public Member Functions

- virtual ~**Queue** ()
- virtual std::string **getQueueName** () const =0 throw (CMSException)

  Gets the name of this queue.

6.443.1 Detailed Description

An interface encapsulating a provider-specific queue name. Messages sent to a **Queue** (p. 2157) are sent to a Single Subscriber on that **Queue** (p. 2157) **Destination** (p. 1190). This allows for Queues to be used as load balances implementing a SEDA based architecture. The length of time that a Provider will store a **Message** (p. 1753) in a **Queue** (p. 2157) is not defined by the CMS API, consult your Provider documentation for this information.

Since:

1.0

6.443.2 Constructor & Destructor Documentation

6.443.2.1 virtual cms::Queue::~Queue () [inline, virtual]

6.443.3 Member Function Documentation

6.443.3.1 virtual std::string cms::Queue::getQueueName () const throw (CMSException) [pure virtual]

Gets the name of this queue.

Returns:

The queue name.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::commands::ActiveMQQueue** (p. 340).

The documentation for this class was generated from the following file:

- src/main/cms/**Queue.h**

6.444 cms::QueueBrowser Class Reference

This class implements in interface for browsing the messages in a **Queue** (p. 2157) without removing them.

`#include <src/main/cms/QueueBrowser.h>`Inheritance diagram for cms::QueueBrowser:

Public Member Functions

- virtual ~**QueueBrowser** ()
- virtual const **Queue** * **getQueue** () const =0 throw (cms::CMSException)
- virtual std::string **getMessageSelector** () const =0 throw (cms::CMSException)
- virtual std::vector< const **cms::Message** * > **getEnumeration** () const =0 throw (cms::CMSException)

  Gets an enumeration for browsing the current queue messages in the order they would be received.

6.444.1 Detailed Description

This class implements in interface for browsing the messages in a **Queue** (p. 2157) without removing them. The `getEnumeration` method of this class returns a static snapshot of the **Queue** (p. 2157) at the time the method is called. Since new Message's can be arriving and old Message's could expire the client should periodically refresh its view by calling `getEnumeration` again.

Since:

1.1

6.444.2 Constructor & Destructor Documentation

6.444.2.1 virtual cms::QueueBrowser::~QueueBrowser () [inline, virtual]

6.444.3 Member Function Documentation

6.444.3.1 virtual std::vector<const cms::Message*> cms::QueueBrowser::getEnumeration () const throw (cms::CMSException) [pure virtual]

Gets an enumeration for browsing the current queue messages in the order they would be received. The enumeration returned is a static view of the **Queue** (p. 2157) and is not updated as new Messages arrive, the client should refresh its enumeration by calling this method again.

Returns:

an STL vector for browsing the messages.

Exceptions:

CMSException *(*p. ***850)*** if an internal error occurs.

6.444.3.2 virtual std::string cms::QueueBrowser::getMessageSelector () const throw (cms::CMSException) [pure virtual]

Returns:

the MessageSelector that is used on when this browser was created or empty string if no selector was present.

Exceptions:

CMSException *(*p. ***850)*** if an internal error occurs.

6.444.3.3 virtual const Queue* cms::QueueBrowser::getQueue () const throw (cms::CMSException) [pure virtual]

Returns:

the **Queue** (p. 2157) that this browser is listening on.

Exceptions:

CMSException *(*p. ***850)*** if an internal error occurs.

The documentation for this class was generated from the following file:

- src/main/cms/**QueueBrowser.h**

6.445 decaf::util::Random Class Reference

Random (p. 2160) Value Generator which is used to generate a stream of pseudorandom numbers.

`#include <src/main/decaf/util/Random.h>`

Public Member Functions

- **Random** ()

  Construct a random generator with the current time of day in milliseconds as the initial state.

- **Random** (unsigned long long seed)

  Construct a random generator with the given `seed` *as the initial state.*

- bool **nextBoolean** ()

  Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

- void **nextBytes** (std::vector< unsigned char > &buf)

  Modifies the byte array by a random sequence of bytes generated by this random number generator.

- double **nextDouble** ()

  Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

- float **nextFloat** ()

  Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

- double **nextGaussian** ()

  Pseudo-randomly generates (approximately) a normally distributed `double` *value with mean 0.0 and a standard deviation value of* `1.0` *using the polar method of G.*

- int **nextInt** ()

  Generates a uniformly distributed 32-bit `int` *value from the this random number sequence.*

- int **nextInt** (int n) throw (lang::exceptions::IllegalArgumentException)

  Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` *(exclusively).*

- long long **nextLong** ()

  Generates a uniformly distributed 64-bit `int` *value from the this random number sequence.*

- void **setSeed** (unsigned long long seed)

  Modifies the seed using linear congruential formula presented in The Art of Computer Programming, Volume 2, Section 3.2.1.

Protected Member Functions

- virtual int **next** (int bits)

  Answers a pseudo-random uniformly distributed `int` *value of the number of bits specified by the argument* `bits` *as described by Donald E.*

6.445.1 Detailed Description

Random (p. 2160) Value Generator which is used to generate a stream of pseudorandom numbers. The algorithms implemented by class **Random** (p. 2160) use a protected utility method that on each invocation can supply up to 32 pseudorandomly generated bits.

Since:

1.0

6.445.2 Constructor & Destructor Documentation

6.445.2.1 decaf::util::Random::Random ()

Construct a random generator with the current time of day in milliseconds as the initial state.

See also:

setSeed (p. 2164)

6.445.2.2 decaf::util::Random::Random (unsigned long long *seed*)

Construct a random generator with the given `seed` as the initial state.

Parameters:

seed the seed that will determine the initial state of this random number generator

See also:

setSeed (p. 2164)

6.445.3 Member Function Documentation

6.445.3.1 virtual int decaf::util::Random::next (int *bits*) [protected, virtual]

Answers a pseudo-random uniformly distributed `int` value of the number of bits specified by the argument `bits` as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.2.1.

Returns:

int a pseudo-random generated int number

Parameters:

bits number of bits of the returned value

See also:

nextBytes (p. 2162)
nextDouble (p. 2162)
nextFloat (p. 2162)
nextInt() (p. 2163)

nextInt(int) (p. 2163)
nextGaussian (p. 2163)
nextLong (p. 2163)

6.445.3.2 bool decaf::util::Random::nextBoolean ()

Answers the next pseudo-random, uniformly distributed boolean value generated by this generator.

Returns:

boolean a pseudo-random, uniformly distributed boolean value

6.445.3.3 void decaf::util::Random::nextBytes (std::vector< unsigned char > & *buf*)

Modifies the byte array by a random sequence of bytes generated by this random number generator.

Parameters:

buf non-null array to contain the new random bytes

See also:

next (p. 2161)

6.445.3.4 double decaf::util::Random::nextDouble ()

Generates a normally distributed random double number between 0.0 inclusively and 1.0 exclusively.

Returns:

double

See also:

nextFloat (p. 2162)

6.445.3.5 float decaf::util::Random::nextFloat ()

Generates a normally distributed random float number between 0.0 inclusively and 1.0 exclusively.

Returns:

float a random float number between 0.0 and 1.0

See also:

nextDouble (p. 2162)

6.445.3.6 double decaf::util::Random::nextGaussian ()

Pseudo-randomly generates (approximately) a normally distributed `double` value with mean 0.0 and a standard deviation value of `1.0` using the *polar method* of G. E. P. Box, M. E. Muller, and G. Marsaglia, as described by Donald E. Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, section 3.4.1, subsection C, algorithm P

Returns:

double

See also:

nextDouble (p. 2162)

6.445.3.7 int decaf::util::Random::nextInt (int *n*) throw (lang::exceptions::IllegalArgumentException)

Returns to the caller a new pseudo-random integer value which is uniformly distributed between 0 (inclusively) and the value of `n` (exclusively).

Returns:

int

Parameters:

n int

Exceptions:

IllegalArgumentException

6.445.3.8 int decaf::util::Random::nextInt ()

Generates a uniformly distributed 32-bit `int` value from the this random number sequence.

Returns:

int uniformly distributed `int` value

See also:

next (p. 2161)
nextLong (p. 2163)

6.445.3.9 long long decaf::util::Random::nextLong ()

Generates a uniformly distributed 64-bit `int` value from the this random number sequence.

Returns:

64-bit `int` random number

See also:

next (p. 2161)
nextInt() (p. 2163)
nextInt(int) (p. 2163)

6.445.3.10 void decaf::util::Random::setSeed (unsigned long long *seed*)

Modifies the seed using linear congruential formula presented in *The Art of Computer Programming, Volume 2*, Section 3.2.1.

Parameters:

seed the seed that alters the state of the random number generator

See also:

next (p. 2161)
Random() (p. 2161)
Random(long)

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Random.h**

6.446 decaf::io::Reader Class Reference

`#include <src/main/decaf/io/Reader.h>`

Public Member Functions

- virtual ~**Reader** ()
- virtual void **setInputStream** (**InputStream** *is)=0

  Sets the target input stream.

- virtual **InputStream** * **getInputStream** ()=0

  Gets the target input stream.

- virtual std::size_t **read** (unsigned char *buffer, std::size_t count)=0 throw (IOException, lang::exceptions::NullPointerException)

  Attempts to read an array of bytes from the stream.

- virtual unsigned char **readByte** ()=0 throw (IOException)

  Attempts to read a byte from the input stream.

6.446.1 Constructor & Destructor Documentation

6.446.1.1 virtual decaf::io::Reader::~Reader () [inline, virtual]

6.446.2 Member Function Documentation

6.446.2.1 virtual InputStream* decaf::io::Reader::getInputStream () [pure virtual]

Gets the target input stream.

6.446.2.2 virtual std::size_t decaf::io::Reader::read (unsigned char * *buffer*, std::size_t *count*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Attempts to read an array of bytes from the stream.

Parameters:

buffer The target byte buffer.

count The number of bytes to read.

Returns:

The number of bytes read.

Exceptions:

IOException *(* p. *1477)* thrown if an error occurs.

6.446.2.3 virtual unsigned char decaf::io::Reader::readByte () throw (IOException) [pure virtual]

Attempts to read a byte from the input stream.

Returns:

The byte.

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

6.446.2.4 virtual void decaf::io::Reader::setInputStream (InputStream * *is*) [pure virtual]

Sets the target input stream.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Reader.h**

6.447 decaf::nio::ReadOnlyBufferException Class Reference

`#include <src/main/decaf/nio/ReadOnlyBufferException.h>`Inheritance diagram for decaf::nio::ReadOnlyBufferException:

Public Member Functions

- **ReadOnlyBufferException** () throw ()
  Default Constructor.
- **ReadOnlyBufferException** (const **lang::Exception** &ex) throw ()
  Copy Constructor.
- **ReadOnlyBufferException** (const **ReadOnlyBufferException** &ex) throw ()
  Copy Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **ReadOnlyBufferException** (const std::exception ***cause**) throw ()
  Constructor.
- **ReadOnlyBufferException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor.
- virtual **ReadOnlyBufferException** * **clone** () const
  Clones this exception.
- virtual ~**ReadOnlyBufferException** () throw ()

6.447.1 Constructor & Destructor Documentation

6.447.1.1 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException () throw () `[inline]`

Default Constructor.

6.447.1.2 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const lang::Exception & *ex*) throw () `[inline]`

Copy Constructor.

Parameters:

ex the exception to copy

6.447.1.3 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const ReadOnlyBufferException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.447.1.4 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.447.1.5 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.447.1.6 decaf::nio::ReadOnlyBufferException::ReadOnlyBufferException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.447.1.7 virtual decaf::nio::ReadOnlyBufferException::~ReadOnlyBufferException () throw () [inline, virtual]

6.447.2 Member Function Documentation

6.447.2.1 virtual ReadOnlyBufferException* decaf::nio::ReadOnlyBufferException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Reimplemented from **decaf::lang::exceptions::UnsupportedOperationException** (p. 2637).

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ReadOnlyBufferException.h**

6.448 decaf::util::concurrent::locks::ReadWriteLock Class Reference

A **ReadWriteLock** (p. 2170) maintains a pair of associated **locks** (p. 119), one for read-only operations and one for writing.

`#include <src/main/decaf/util/concurrent/locks/ReadWriteLock.h>`

Public Member Functions

- virtual ~**ReadWriteLock** ()
- virtual **Lock** & **readLock** ()=0

  Returns the lock used for reading.

- virtual **Lock** & **writeLock** ()=0

  Returns the lock used for writing.

6.448.1 Detailed Description

A **ReadWriteLock** (p. 2170) maintains a pair of associated **locks** (p. 119), one for read-only operations and one for writing. The read lock may be held simultaneously by multiple reader threads, so long as there are no writers. The write lock is exclusive.

All **ReadWriteLock** (p. 2170) implementations must guarantee that the memory synchronization effects of writeLock operations (as specified in the **Lock** (p. 1618) interface) also hold with respect to the associated readLock. That is, a thread successfully acquiring the read lock will see all updates made upon previous release of the write lock.

A read-write lock allows for a greater level of concurrency in accessing shared data than that permitted by a mutual exclusion lock. It exploits the fact that while only a single thread at a time (a writer thread) can modify the shared data, in many cases any number of threads can concurrently read the data (hence reader threads). In theory, the increase in concurrency permitted by the use of a read-write lock will lead to performance improvements over the use of a mutual exclusion lock. In practice this increase in concurrency will only be fully realized on a multi-processor, and then only if the access patterns for the shared data are suitable.

Whether or not a read-write lock will improve performance over the use of a mutual exclusion lock depends on the frequency that the data is read compared to being modified, the duration of the read and write operations, and the contention for the data - that is, the number of threads that will try to read or write the data at the same time. For example, a collection that is initially populated with data and thereafter infrequently modified, while being frequently searched (such as a directory of some kind) is an ideal candidate for the use of a read-write lock. However, if updates become frequent then the data spends most of its time being exclusively locked and there is little, if any increase in concurrency. Further, if the read operations are too short the overhead of the read-write lock implementation (which is inherently more complex than a mutual exclusion lock) can dominate the execution cost, particularly as many read-write lock implementations still serialize all threads through a small section of code. Ultimately, only profiling and measurement will establish whether the use of a read-write lock is suitable for your application.

Although the basic operation of a read-write lock is straight-forward, there are many policy decisions that an implementation must make, which may affect the effectiveness of the read-write lock in a given application. Examples of these policies include:

∗ Determining whether to grant the read lock or the write lock, when both readers and writers are waiting, at the time that a writer releases the write lock. Writer preference is common, as writes are expected to be short and infrequent. Reader preference is less common as it can lead to lengthy delays for a write if the readers are frequent and long-lived as expected. Fair, or "in-order" implementations are also possible. ∗ Determining whether readers that request the read lock while a reader is active and a writer is waiting, are granted the read lock. Preference to the reader can delay the writer indefinitely, while preference to the writer can reduce the potential for concurrency. ∗ Determining whether the **locks** (p. 119) are reentrant: can a thread with the write lock reacquire it? Can it acquire a read lock while holding the write lock? Is the read lock itself reentrant? ∗ Can the write lock be downgraded to a read lock without allowing an intervening writer? Can a read lock be upgraded to a write lock, in preference to other waiting readers or writers?

You should consider all of these things when evaluating the suitability of a given implementation for your application.

Since:

1.0

6.448.2 Constructor & Destructor Documentation

6.448.2.1 virtual decaf::util::concurrent::locks::ReadWriteLock::~ReadWriteLock () [inline, virtual]

6.448.3 Member Function Documentation

6.448.3.1 virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::readLock () [pure virtual]

Returns the lock used for reading.

Returns:

the lock used for reading.

6.448.3.2 virtual Lock& decaf::util::concurrent::locks::ReadWriteLock::writeLock () [pure virtual]

Returns the lock used for writing.

Returns:

the lock used for writing.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/locks/**ReadWriteLock.h**

6.449 activemq::cmsutil::CmsTemplate::ReceiveExecutor Class Reference

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate::ReceiveExecutor:

Public Member Functions

- **ReceiveExecutor** (**CmsTemplate** *parent, **cms::Destination** *destination, const std::string &**selector**, bool **noLocal**)
- virtual ~**ReceiveExecutor** ()
- virtual void **doInCms** (**cms::Session** *session) throw (cms::CMSException)

  Execute any number of operations against the supplied CMS session.

- virtual **cms::Destination** * **getDestination** (**cms::Session** *session AMQCPP_UNUSED) throw (cms::CMSException)
- **cms::Message** * **getMessage** ()

Protected Attributes

- **cms::Destination** * **destination**
- std::string **selector**
- bool **noLocal**
- **cms::Message** * **message**
- **CmsTemplate** * **parent**

6.449.1 Constructor & Destructor Documentation

6.449.1.1 activemq::cmsutil::CmsTemplate::ReceiveExecutor::ReceiveExecutor (CmsTemplate * *parent*, cms::Destination * *destination*, const std::string & *selector*, bool *noLocal*) [inline]

6.449.1.2 virtual activemq::cmsutil::CmsTemplate::ReceiveExecutor::~ReceiveExecutor () [inline, virtual]

6.449.2 Member Function Documentation

6.449.2.1 virtual void activemq::cmsutil::CmsTemplate::ReceiveExecutor::doInCms (cms::Session * *session*) throw (cms::CMSException) [virtual]

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS `Session`

Exceptions:

cms::CMSException *(*p. ***850****)* if thrown by CMS API methods

Implements **activemq::cmsutil::SessionCallback** (p. 2283).

6.449.2.2 virtual cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getDestination (cms::Session *session *AMQCPP_UNUSED*) throw (cms::CMSException) [inline, virtual]

6.449.2.3 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::getMessage () [inline]

6.449.3 Field Documentation

6.449.3.1 cms::Destination* activemq::cmsutil::CmsTemplate::ReceiveExecutor::destination [protected]

6.449.3.2 cms::Message* activemq::cmsutil::CmsTemplate::ReceiveExecutor::message [protected]

6.449.3.3 bool activemq::cmsutil::CmsTemplate::ReceiveExecutor::noLocal [protected]

6.449.3.4 CmsTemplate* activemq::cmsutil::CmsTemplate::ReceiveExecutor::parent [protected]

6.449.3.5 std::string activemq::cmsutil::CmsTemplate::ReceiveExecutor::selector [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.450 decaf::util::concurrent::RejectedExecutionException Class Reference

`#include <src/main/decaf/util/concurrent/RejectedExecutionException.h>`Inheritance diagram for decaf::util::concurrent::RejectedExecutionException:

Public Member Functions

- **RejectedExecutionException** () throw ()
  Default Constructor.
- **RejectedExecutionException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **RejectedExecutionException** (const **RejectedExecutionException** &ex) throw ()
  Copy Constructor.
- **RejectedExecutionException** (const std::exception *__cause__) throw ()
  Constructor.
- **RejectedExecutionException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **RejectedExecutionException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **RejectedExecutionException** * **clone** () const
  Clones this exception.
- virtual ~**RejectedExecutionException** () throw ()

6.450.1 Constructor & Destructor Documentation

6.450.1.1 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException () throw () [inline]

Default Constructor.

6.450.1.2 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.450.1.3 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const RejectedExecutionException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex - The Exception to copy in this new instance.

References decaf::lang::Exception::Exception().

6.450.1.4 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.450.1.5 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

msg - The message to report

... - list of primitives that are formatted into the message

6.450.1.6 decaf::util::concurrent::RejectedExecutionException::RejectedExecutionException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file - The file name where exception occurs

lineNumber - The line number where the exception occurred.

cause - The exception that was the cause for this one to be thrown.

msg - The message to report

... - list of primitives that are formatted into the message

6.450.1.7 virtual decaf::util::concurrent::RejectedExecutionException::~RejectedExecutionException () throw () [inline, virtual]

6.450.2 Member Function Documentation

6.450.2.1 virtual RejectedExecutionException* decaf::util::concurrent::RejectedExecutionException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new instance this exception type with a copy the current state.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**RejectedExecutionException.h**

6.451 activemq::commands::RemoveInfo Class Reference

`#include <src/main/activemq/commands/RemoveInfo.h>`Inheritance diagram for activemq::commands::RemoveInfo:

Public Member Functions

- **RemoveInfo** ()
- virtual ~**RemoveInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **RemoveInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **DataStructure** > & **getObjectId** () const
- virtual **Pointer**< **DataStructure** > & **getObjectId** ()
- virtual void **setObjectId** (const **Pointer**< **DataStructure** > &**objectId**)
- virtual bool **isRemoveInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVEINFO** = 12

Protected Member Functions

- **RemoveInfo** (const **RemoveInfo** &)
- **RemoveInfo** & **operator**= (const **RemoveInfo** &)

Protected Attributes

- **Pointer**< **DataStructure** > **objectId**

6.451.1 Constructor & Destructor Documentation

6.451.1.1 activemq::commands::RemoveInfo::RemoveInfo (const RemoveInfo &) [inline, protected]

6.451.1.2 activemq::commands::RemoveInfo::RemoveInfo ()

6.451.1.3 virtual activemq::commands::RemoveInfo::~RemoveInfo () [virtual]

6.451.2 Member Function Documentation

6.451.2.1 virtual RemoveInfo* activemq::commands::RemoveInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.451.2.2 virtual void activemq::commands::RemoveInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.451.2.3 virtual bool activemq::commands::RemoveInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.451.2.4 virtual unsigned char activemq::commands::RemoveInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.451.2.5 virtual Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId () [virtual]

6.451.2.6 virtual const Pointer<DataStructure>& activemq::commands::RemoveInfo::getObjectId () const [virtual]

6.451.2.7 virtual bool activemq::commands::RemoveInfo::isRemoveInfo () const [inline, virtual]

Returns:

an answer of true to the **isRemoveInfo()** (p. 2179) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.451.2.8 RemoveInfo& activemq::commands::RemoveInfo::operator= (const RemoveInfo &) [inline, protected]

6.451.2.9 virtual void activemq::commands::RemoveInfo::setObjectId (const Pointer< DataStructure > & *objectId*) [virtual]

6.451.2.10 virtual std::string activemq::commands::RemoveInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.451.2.11 virtual Pointer<Command> activemq::commands::RemoveInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.451.3 Field Documentation

6.451.3.1 const unsigned char activemq::commands::RemoveInfo::ID_REMOVEINFO = 12 [static]

6.451.3.2 Pointer<DataStructure> activemq::commands::RemoveInfo::objectId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveInfo.h**

6.452 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshall Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2181).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.452.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2181). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.452.2 Constructor & Destructor Documentation

6.452.2.1 activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::RemoveInfoMarshall () [inline]

6.452.2.2 virtual activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::~RemoveInfoMarsha () [inline, virtual]

6.452.3 Member Function Documentation

6.452.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.452.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.452.3.3 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.452.3.4 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.452.3.5 virtual int activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.452.3.6 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.452.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveInfoMarshaller.h**

6.453 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshal Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2185).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.453.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2185). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.453.2 Constructor & Destructor Documentation

6.453.2.1 activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::RemoveInfoMarshall () [inline]

6.453.2.2 virtual activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::~RemoveInfoMarsha () [inline, virtual]

6.453.3 Member Function Documentation

6.453.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.453.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.453.3.3 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.453.3.4 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.453.3.5 virtual int activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.453.3.6 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.453.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveInfoMarshaller.h**

6.454 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshall Class Reference

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2189).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller:

Public Member Functions

- **RemoveInfoMarshaller** ()
- virtual ~**RemoveInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.454.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveInfoMarshaller** (p. 2189). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.454.2 Constructor & Destructor Documentation

6.454.2.1 activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::RemoveInfoMarshall () [inline]

6.454.2.2 virtual activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::~RemoveInfoMarsha () [inline, virtual]

6.454.3 Member Function Documentation

6.454.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.454.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.454.3.3 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.454.3.4 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.454.3.5 virtual int activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.454.3.6 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.454.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveInfoMarshaller.h**

6.455 activemq::commands::RemoveSubscriptionInfo Class Reference

`#include <src/main/activemq/commands/RemoveSubscriptionInfo.h>`Inheritance diagram for activemq::commands::RemoveSubscriptionInfo:

Public Member Functions

- **RemoveSubscriptionInfo** ()
- virtual ~**RemoveSubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **RemoveSubscriptionInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &**connectionId**)
- virtual const std::string & **getSubcriptionName** () const
- virtual std::string & **getSubcriptionName** ()
- virtual void **setSubcriptionName** (const std::string &**subcriptionName**)
- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &**clientId**)
- virtual bool **isRemoveSubscriptionInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REMOVESUBSCRIPTIONINFO** = 9

Protected Member Functions

- **RemoveSubscriptionInfo** (const **RemoveSubscriptionInfo** &)
- **RemoveSubscriptionInfo** & **operator=** (const **RemoveSubscriptionInfo** &)

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- std::string **subcriptionName**
- std::string **clientId**

6.455.1 Constructor & Destructor Documentation

6.455.1.1 activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo (const RemoveSubscriptionInfo &) [inline, protected]

6.455.1.2 activemq::commands::RemoveSubscriptionInfo::RemoveSubscriptionInfo ()

6.455.1.3 virtual activemq::commands::RemoveSubscriptionInfo::~RemoveSubscriptionInfo () [virtual]

6.455.2 Member Function Documentation

6.455.2.1 virtual RemoveSubscriptionInfo* activemq::commands::RemoveSubscriptionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.455.2.2 virtual void activemq::commands::RemoveSubscriptionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.455.2.3 virtual bool activemq::commands::RemoveSubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.455.2.4 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () [virtual]

6.455.2.5 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getClientId () const [virtual]

6.455.2.6 virtual Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () [virtual]

6.455.2.7 virtual const Pointer<ConnectionId>& activemq::commands::RemoveSubscriptionInfo::getConnectionId () const [virtual]

6.455.2.8 virtual unsigned char activemq::commands::RemoveSubscriptionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.455.2.9 virtual std::string& activemq::commands::RemoveSubscriptionInfo::getSubcriptionName () [virtual]

6.455.2.10 virtual const std::string& activemq::commands::RemoveSubscriptionInfo::getSubcriptionName () const [virtual]

6.455.2.11 virtual bool activemq::commands::RemoveSubscriptionInfo::isRemoveSubscriptionInfo () const [inline, virtual]

Returns:

an answer of true to the **isRemoveSubscriptionInfo()** (p. 2196) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.455.2.12 RemoveSubscriptionInfo& activemq::commands::RemoveSubscriptionInfo::operator= (const RemoveSubscriptionInfo &) [inline, protected]

6.455.2.13 virtual void activemq::commands::RemoveSubscriptionInfo::setClientId (const std::string & *clientId*) [virtual]

6.455.2.14 virtual void activemq::commands::RemoveSubscriptionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.455.2.15 virtual void activemq::commands::RemoveSubscriptionInfo::setSubcriptionName (const std::string & *subcriptionName*) [virtual]

6.455.2.16 virtual std::string activemq::commands::RemoveSubscriptionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.455.2.17 virtual Pointer<Command> activemq::commands::RemoveSubscriptionInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.455.3 Field Documentation

6.455.3.1 std::string activemq::commands::RemoveSubscriptionInfo::clientId [protected]

6.455.3.2 Pointer<ConnectionId> activemq::commands::RemoveSubscriptionInfo::connectionId [protected]

6.455.3.3 const unsigned char activemq::commands::RemoveSubscriptionInfo::ID_REMOVESUBSCRIPTIONINFO = 9 [static]

6.455.3.4 std::string activemq::commands::RemoveSubscriptionInfo::subcriptionName [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**RemoveSubscriptionInfo.h**

6.456 activemq::wireformat::openwire::marshal::v2::RemoveSubscription Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2198).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h>`In diagram for activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.456.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2198). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.456.2 Constructor & Destructor Documentation

6.456.2.1 activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::Remove () [inline]

6.456.2.2 virtual activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::~Remov () [inline, virtual]

6.456.3 Member Function Documentation

6.456.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.456.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.456.3.3 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.456.3.4 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.456.3.5 virtual int activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.456.3.6 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.456.3.7 virtual void activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**RemoveSubscriptionInfoMarshaller.h**

6.457 activemq::wireformat::openwire::marshal::v3::RemoveSubscription Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2202).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/RemoveSubscriptionInfoMarshaller.h>`In diagram for activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.457.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2202). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.457.2 Constructor & Destructor Documentation

6.457.2.1 activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::Remove () [inline]

6.457.2.2 virtual activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::~Remov () [inline, virtual]

6.457.3 Member Function Documentation

6.457.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::createObj () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.457.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::getDataSt () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.457.3.3 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.457.3.4 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.457.3.5 virtual int activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.457.3.6 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.457.3.7 virtual void activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**RemoveSubscriptionInfoMarshaller.h**

6.458 activemq::wireformat::openwire::marshal::v1::RemoveSubscription Class Reference

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2206).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/RemoveSubscriptionInfoMarshaller.h>`In diagram for activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller:

Public Member Functions

- **RemoveSubscriptionInfoMarshaller** ()
- virtual ~**RemoveSubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.458.1 Detailed Description

Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2206). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.458.2 Constructor & Destructor Documentation

6.458.2.1 activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::Remove

() [inline]

6.458.2.2 virtual activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::~Remov

() [inline, virtual]

6.458.3 Member Function Documentation

6.458.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::createObj

() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.458.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::getDataSt

() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.458.3.3 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseMars

(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.458.3.4 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::looseUnm (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.458.3.5 virtual int activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.458.3.6 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightMars (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.458.3.7 virtual void activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller::tightUnma (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**RemoveSubscriptionInfoMarshaller.h**

6.459 activemq::commands::ReplayCommand Class Reference

`#include <src/main/activemq/commands/ReplayCommand.h>`Inheritance diagram for activemq::commands::ReplayCommand:

Public Member Functions

- **ReplayCommand** ()
- virtual ~**ReplayCommand** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ReplayCommand** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual int **getFirstNakNumber** () const
- virtual void **setFirstNakNumber** (int **firstNakNumber**)
- virtual int **getLastNakNumber** () const
- virtual void **setLastNakNumber** (int **lastNakNumber**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_REPLAYCOMMAND** = 65

Protected Member Functions

- **ReplayCommand** (const **ReplayCommand** &)
- **ReplayCommand** & **operator**= (const **ReplayCommand** &)

Protected Attributes

- int **firstNakNumber**
- int **lastNakNumber**

6.459.1 Constructor & Destructor Documentation

6.459.1.1 activemq::commands::ReplayCommand::ReplayCommand (const ReplayCommand &) [inline, protected]

6.459.1.2 activemq::commands::ReplayCommand::ReplayCommand ()

6.459.1.3 virtual activemq::commands::ReplayCommand::~ReplayCommand () [virtual]

6.459.2 Member Function Documentation

6.459.2.1 virtual ReplayCommand* activemq::commands::ReplayCommand::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.459.2.2 virtual void activemq::commands::ReplayCommand::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.459.2.3 virtual bool activemq::commands::ReplayCommand::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.459.2.4 virtual unsigned char activemq::commands::ReplayCommand::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.459.2.5 virtual int activemq::commands::ReplayCommand::getFirstNakNumber () const [virtual]

6.459.2.6 virtual int activemq::commands::ReplayCommand::getLastNakNumber () const [virtual]

6.459.2.7 ReplayCommand& activemq::commands::ReplayCommand::operator= (const ReplayCommand &) [inline, protected]

6.459.2.8 virtual void activemq::commands::ReplayCommand::setFirstNakNumber (int *firstNakNumber*) [virtual]

6.459.2.9 virtual void activemq::commands::ReplayCommand::setLastNakNumber (int *lastNakNumber*) [virtual]

6.459.2.10 virtual std::string activemq::commands::ReplayCommand::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.459.2.11 virtual Pointer<Command> activemq::commands::ReplayCommand::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.459.3 Field Documentation

6.459.3.1 int activemq::commands::ReplayCommand::firstNakNumber [protected]

6.459.3.2 const unsigned char activemq::commands::ReplayCommand::ID_REPLAYCOMMAND = 65 [static]

6.459.3.3 int activemq::commands::ReplayCommand::lastNakNumber [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ReplayCommand.h**

6.460 activemq::wireformat::openwire::marshal::v2::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2214).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ReplayCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.460.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2214). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.460.2 Constructor & Destructor Documentation

6.460.2.1 activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::ReplayComma () [inline]

6.460.2.2 virtual activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::~ReplayComm () [inline, virtual]

6.460.3 Member Function Documentation

6.460.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.460.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.460.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.460.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.460.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.460.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.460.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ReplayCommandMarshaller.h**

6.461 activemq::wireformat::openwire::marshal::v3::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2218).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.461.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2218). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.461.2 Constructor & Destructor Documentation

6.461.2.1 activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::ReplayComma () [inline]

6.461.2.2 virtual activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::~ReplayComm () [inline, virtual]

6.461.3 Member Function Documentation

6.461.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.461.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.461.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.461.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.461.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.461.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.461.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ReplayCommandMarshaller.h**

6.462 activemq::wireformat::openwire::marshal::v1::ReplayCommandMa Class Reference

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2222).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller:

Public Member Functions

- **ReplayCommandMarshaller** ()
- virtual ~**ReplayCommandMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.462.1 Detailed Description

Marshaling code for Open Wire Format for **ReplayCommandMarshaller** (p. 2222). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.462.2 Constructor & Destructor Documentation

6.462.2.1 activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::ReplayComma () [inline]

6.462.2.2 virtual activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::~ReplayComm () [inline, virtual]

6.462.3 Member Function Documentation

6.462.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.462.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.462.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.462.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.462.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.462.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.462.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ReplayCommandMarshaller.h**

6.463 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor Class Reference

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveProducerExecutor:

Public Member Functions

- **ResolveProducerExecutor** (**ProducerCallback** ∗**action**, **CmsTemplate** ∗**parent**, const std::string &destinationName)
- virtual ∼**ResolveProducerExecutor** ()
- virtual **cms::Destination** ∗ **getDestination** (**cms::Session** ∗session) throw (cms::CMSException)

6.463.1 Constructor & Destructor Documentation

6.463.1.1 activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::ResolveProducerExecutor (ProducerCallback ∗ *action*, CmsTemplate ∗ *parent*, const std::string & *destinationName*) `[inline]`

6.463.1.2 virtual activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::∼ResolveProducerExecutor () `[inline, virtual]`

6.463.2 Member Function Documentation

6.463.2.1 virtual cms::Destination∗ activemq::cmsutil::CmsTemplate::ResolveProducerExecutor::getDestination (cms::Session ∗ *session*) throw (cms::CMSException) `[virtual]`

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.464 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor Class Reference

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor:

Public Member Functions

- **ResolveReceiveExecutor** (**CmsTemplate** ∗**parent**, const std::string &**selector**, bool **noLocal**, const std::string &destinationName)
- virtual ∼**ResolveReceiveExecutor** ()
- virtual **cms::Destination** ∗ **getDestination** (**cms::Session** ∗session) throw (cms::CMSException)

6.464.1 Constructor & Destructor Documentation

6.464.1.1 activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::ResolveReceiveExecutor (CmsTemplate ∗ *parent*, const std::string & *selector*, bool *noLocal*, const std::string & *destinationName*) `[inline]`

6.464.1.2 virtual activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::∼ResolveReceiveExecutor () `[inline, virtual]`

6.464.2 Member Function Documentation

6.464.2.1 virtual cms::Destination∗ activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor::getDestination (cms::Session ∗ *session*) throw (cms::CMSException) `[virtual]`

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.465 activemq::cmsutil::ResourceLifecycleManager Class Reference

Manages the lifecycle of a set of CMS resources.

`#include <src/main/activemq/cmsutil/ResourceLifecycleManager.h>`

Public Member Functions

- **ResourceLifecycleManager** ()
- virtual ~**ResourceLifecycleManager** ()

  Destructor - calls `destroy`.
- void **addConnection** (**cms::Connection** *connection)

  Adds a connection so that its life will be managed by this object.
- void **addSession** (**cms::Session** *session)

  Adds a session so that its life will be managed by this object.
- void **addDestination** (**cms::Destination** *dest)

  Adds a destination so that its life will be managed by this object.
- void **addMessageProducer** (**cms::MessageProducer** *producer)

  Adds a message producer so that its life will be managed by this object.
- void **addMessageConsumer** (**cms::MessageConsumer** *consumer)

  Adds a message consumer so that its life will be managed by this object.
- void **destroy** () throw (cms::CMSException)

  Closes and destroys the contained CMS resources.
- void **releaseAll** ()

  Releases all of the contained resources so that this object will no longer control their lifetimes.

6.465.1 Detailed Description

Manages the lifecycle of a set of CMS resources. A call to `destroy` will close and destroy all of the contained resources in the appropriate manner.

6.465.2 Constructor & Destructor Documentation

6.465.2.1 activemq::cmsutil::ResourceLifecycleManager::ResourceLifecycleManager ()

6.465.2.2 virtual activemq::cmsutil::ResourceLifecycleManager::~ResourceLifecycleManager () `[virtual]`

Destructor - calls `destroy`.

6.465.3 Member Function Documentation

6.465.3.1 void activemq::cmsutil::ResourceLifecycleManager::addConnection (cms::Connection * *connection*) [inline]

Adds a connection so that its life will be managed by this object.

Parameters:

connection the object to be managed

6.465.3.2 void activemq::cmsutil::ResourceLifecycleManager::addDestination (cms::Destination * *dest*) [inline]

Adds a destination so that its life will be managed by this object.

Parameters:

dest the object to be managed

6.465.3.3 void activemq::cmsutil::ResourceLifecycleManager::addMessageConsumer (cms::MessageConsumer * *consumer*) [inline]

Adds a message consumer so that its life will be managed by this object.

Parameters:

consumer the object to be managed

6.465.3.4 void activemq::cmsutil::ResourceLifecycleManager::addMessageProducer (cms::MessageProducer * *producer*) [inline]

Adds a message producer so that its life will be managed by this object.

Parameters:

producer the object to be managed

6.465.3.5 void activemq::cmsutil::ResourceLifecycleManager::addSession (cms::Session * *session*) [inline]

Adds a session so that its life will be managed by this object.

Parameters:

session the object to be managed

6.465.3.6 void activemq::cmsutil::ResourceLifecycleManager::destroy () throw (cms::CMSException)

Closes and destroys the contained CMS resources.

Exceptions:

cms::CMSException (p. ***850***) thrown if an error occurs.

6.465.3.7 void activemq::cmsutil::ResourceLifecycleManager::releaseAll ()

Releases all of the contained resources so that this object will no longer control their lifetimes.

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**ResourceLifecycleManager.h**

6.466 activemq::commands::Response Class Reference

`#include <src/main/activemq/commands/Response.h>`Inheritance diagram for activemq::commands::Response:

Public Member Functions

- **Response** ()
- virtual ~**Response** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **Response** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual int **getCorrelationId** () const
- virtual void **setCorrelationId** (int **correlationId**)
- virtual bool **isResponse** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_RESPONSE** = 30

Protected Member Functions

- **Response** (const **Response** &)
- **Response** & **operator**= (const **Response** &)

Protected Attributes

- int **correlationId**

6.466.1 Constructor & Destructor Documentation

6.466.1.1 activemq::commands::Response::Response (const Response &) [inline, protected]

6.466.1.2 activemq::commands::Response::Response ()

6.466.1.3 virtual activemq::commands::Response::~Response () [virtual]

6.466.2 Member Function Documentation

6.466.2.1 virtual Response* activemq::commands::Response::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::ExceptionResponse** (p. 1277), and **activemq::commands::IntegerResponse** (p. 1443).

6.466.2.2 virtual void activemq::commands::Response::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::ExceptionResponse** (p. 1277), and **activemq::commands::IntegerResponse** (p. 1443).

6.466.2.3 virtual bool activemq::commands::Response::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1102), **activemq::commands::DataResponse** (p. 1133), **activemq::commands::ExceptionResponse** (p. 1277), and **activemq::commands::IntegerResponse** (p. 1443).

6.466.2.4 virtual int activemq::commands::Response::getCorrelationId () const [virtual]

6.466.2.5 virtual unsigned char activemq::commands::Response::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::ExceptionResponse** (p. 1277), and **activemq::commands::IntegerResponse** (p. 1443).

6.466.2.6 virtual bool activemq::commands::Response::isResponse () const [inline, virtual]

Returns:

an answer of true to the **isResponse()** (p. 2233) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.466.2.7 Response& activemq::commands::Response::operator= (const Response &) [inline, protected]

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::ExceptionResponse** (p. 1278), and **activemq::commands::IntegerResponse** (p. 1444).

6.466.2.8 virtual void activemq::commands::Response::setCorrelationId (int *correlationId*) [virtual]

6.466.2.9 virtual std::string activemq::commands::Response::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

Reimplemented in **activemq::commands::DataArrayResponse** (p. 1103), **activemq::commands::DataResponse** (p. 1134), **activemq::commands::ExceptionResponse** (p. 1278), and **activemq::commands::IntegerResponse** (p. 1444).

6.466.2.10 virtual Pointer<Command> activemq::commands::Response::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.466.3 Field Documentation

6.466.3.1 int activemq::commands::Response::correlationId [protected]

6.466.3.2 const unsigned char activemq::commands::Response::ID_RESPONSE = 30 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**Response.h**

6.467 activemq::transport::mock::ResponseBuilder Class Reference

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

`#include <src/main/activemq/transport/mock/ResponseBuilder.h>`Inheritance diagram for activemq::transport::mock::ResponseBuilder:

Public Member Functions

- virtual ~**ResponseBuilder** ()
- virtual **Pointer**< **Response** > **buildResponse** (const **Pointer**< **Command** > &command)=0

  Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

- virtual void **buildIncomingCommands** (const **Pointer**< **Command** > &command, **decaf::util::StlQueue**< **Pointer**< **Command** > > &queue)=0

  *When called the **ResponseBuilder** (p. 2235) must construct all the Responses or Asynchronous **commands** (p. 59) that would be sent to this client by the Broker upon receipt of the passed command.*

6.467.1 Detailed Description

Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

6.467.2 Constructor & Destructor Documentation

6.467.2.1 virtual activemq::transport::mock::ResponseBuilder::~ResponseBuilder () `[inline, virtual]`

6.467.3 Member Function Documentation

6.467.3.1 virtual void activemq::transport::mock::ResponseBuilder::buildIncomingCommands (const Pointer< Command > & *command*, decaf::util::StlQueue< Pointer< Command > > & *queue*) `[pure virtual]`

When called the **ResponseBuilder** (p. 2235) must construct all the Responses or Asynchronous **commands** (p. 59) that would be sent to this client by the Broker upon receipt of the passed command.

Parameters:

command - The Command being sent to the Broker.

queue - Queue of Command sent back from the broker.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 1988).

6.467.3.2 virtual Pointer<Response> activemq::transport::mock::ResponseBuilder::buildResponse (const Pointer< Command > & *command*) [pure virtual]

Given a Command, check if it requires a response and return the appropriate Response that the Broker would send for this Command.

Parameters:

command - The command to build a response for

Returns:

A Response object pointer, or NULL if no response.

Implemented in **activemq::wireformat::openwire::OpenWireResponseBuilder** (p. 1989).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/mock/**ResponseBuilder.h**

6.468 activemq::transport::correlator::ResponseCorrelator Class Reference

This type of **transport** (p. 67) filter is responsible for correlating asynchronous responses with requests.

`#include <src/main/activemq/transport/correlator/ResponseCorrelator.h>`Inheritance diagram for activemq::transport::correlator::ResponseCorrelator:

Public Member Functions

- **ResponseCorrelator** (const **Pointer**< **Transport** > &**next**)

  Constructor.

- virtual ~**ResponseCorrelator** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given request to the server and waits for the response.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given request to the server and waits for the response.

- virtual void **onCommand** (const **Pointer**< **Command** > &command)

  This is called in the context of the nested transport's reading thread.

- virtual void **start** () throw (cms::CMSException)

  *Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59).*

- virtual void **close** () throw (cms::CMSException)

  Stops the polling thread and closes the streams.

- virtual void **onTransportException** (**Transport** *source, const **decaf::lang::Exception** &ex)

  *Event handler for an exception from a command **transport** (p. 67).*

6.468.1 Detailed Description

This type of **transport** (p. 67) filter is responsible for correlating asynchronous responses with requests. Non-response messages are simply sent directly to the CommandListener. It owns the **transport** (p. 67) that it

6.468.2 Constructor & Destructor Documentation

6.468.2.1 activemq::transport::correlator::ResponseCorrelator::ResponseCorrelator (const Pointer< Transport > & *next*)

Constructor.

Parameters:

next the next **transport** (p. 67) in the chain

6.468.2.2 virtual activemq::transport::correlator::ResponseCorrelator::~ResponseCorrelator () [virtual]

6.468.3 Member Function Documentation

6.468.3.1 virtual void activemq::transport::correlator::ResponseCorrelator::close () throw (cms::CMSException) [virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

CMSException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 2618).

6.468.3.2 virtual void activemq::transport::correlator::ResponseCorrelator::onCommand (const Pointer< Command > & *command*) [virtual]

This is called in the context of the nested transport's reading thread. In the case of a response object, updates the request map and notifies those waiting on the response. Non-response messages are just delegated to the command listener.

Parameters:

command the received from the nested **transport** (p. 67).

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.468.3.3 virtual void activemq::transport::correlator::ResponseCorrelator::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Reimplemented from **activemq::transport::TransportFilter** (p. 2620).

6.468.3.4 virtual void activemq::transport::correlator::ResponseCorrelator::onTransportException (Transport * *source*, const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

source The source of the exception

ex The exception.

6.468.3.5 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response.

Parameters:

command The request to send.

timeout The time to wait for a response.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 2621).

6.468.3.6 virtual Pointer<Response> activemq::transport::correlator::ResponseCorrelator::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Sends the given request to the server and waits for the response.

Parameters:

command The request to send.

Returns:

the response from the server.

Exceptions:

IOException if an error occurs with the request.

Reimplemented from **activemq::transport::TransportFilter** (p. 2622).

6.468.3.7 virtual void activemq::transport::correlator::ResponseCorrelator::start () throw (cms::CMSException) [virtual]

Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this **transport** (p. 67) has already been closed.

Reimplemented from **activemq::transport::TransportFilter** (p. 2622).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/correlator/**ResponseCorrelator.h**

6.469 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2241).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual ~**ResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.469.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2241). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.469.2 Constructor & Destructor Documentation

6.469.2.1 activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::ResponseMarshaller () [inline]

6.469.2.2 virtual activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::~ResponseMarshaller () [inline, virtual]

6.469.3 Member Function Documentation

6.469.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1113), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1144), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1288), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1446).

6.469.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1113), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1144), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1288), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1446).

6.469.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1113), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1144), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1288), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1446).

6.469.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1114), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1145), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1289), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1447).

6.469.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1114), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1145), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1289), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1447).

6.469.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1115), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1290), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1448).

6.469.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller** (p. 1115), **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller** (p. 1146), **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller** (p. 1290), and **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller** (p. 1448).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ResponseMarshaller.h**

6.470 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2246).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual ~**ResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.470.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2246). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.470.2 Constructor & Destructor Documentation

6.470.2.1 activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::ResponseMarshaller () [inline]

6.470.2.2 virtual activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::~ResponseMarshaller () [inline, virtual]

6.470.3 Member Function Documentation

6.470.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454).

6.470.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454).

6.470.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1105), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1136), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1280), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1454).

6.470.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1106), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1137), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1281), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1455).

6.470.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1106), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1137), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1281), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1455).

6.470.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1107), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1138), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1282), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1456).

6.470.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller** (p. 1107), **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller** (p. 1138), **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller** (p. 1282), and **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller** (p. 1456).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ResponseMarshaller.h**

6.471 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller Class Reference

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2251).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ResponseMarshaller:

Public Member Functions

- **ResponseMarshaller** ()
- virtual ~**ResponseMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.471.1 Detailed Description

Marshaling code for Open Wire Format for **ResponseMarshaller** (p. 2251). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.471.2 Constructor & Destructor Documentation

6.471.2.1 activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::ResponseMarshaller () [inline]

6.471.2.2 virtual activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::~ResponseMarshaller () [inline, virtual]

6.471.3 Member Function Documentation

6.471.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450).

6.471.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450).

6.471.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1109), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1140), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1284), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1450).

6.471.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1110), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1141), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1285), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1451).

6.471.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1110), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1141), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1285), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1451).

6.471.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452).

6.471.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ResponseMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller** (p. 1111), **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller** (p. 1142), **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller** (p. 1286), and **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller** (p. 1452).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ResponseMarshaller.h**

6.472 decaf::lang::Runnable Class Reference

Interface for a runnable object - defines a task that can be run by a thread.

`#include <src/main/decaf/lang/Runnable.h>`Inheritance diagram for decaf::lang::Runnable:

Public Member Functions

- virtual ~**Runnable** ()
- virtual void **run** ()=0

  *Run method - called by the **Thread** (p. 2544) class in the context of the thread.*

6.472.1 Detailed Description

Interface for a runnable object - defines a task that can be run by a thread.

6.472.2 Constructor & Destructor Documentation

6.472.2.1 virtual decaf::lang::Runnable::~Runnable () [inline, virtual]

6.472.3 Member Function Documentation

6.472.3.1 virtual void decaf::lang::Runnable::run () [pure virtual]

Run method - called by the **Thread** (p. 2544) class in the context of the thread.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 909), **activemq::threads::DedicatedTaskRunner** (p. 1183), **activemq::transport::IOTransport** (p. 1485), **activemq::transport::mock::InternalCommandListener** (p. 1458), **decaf::lang::Thread** (p. 2545), and **decaf::util::concurrent::PooledThread** (p. 2033).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runnable.h**

6.473 decaf::lang::Runtime Class Reference

`#include <src/main/decaf/lang/Runtime.h>`Inheritance diagram for decaf::lang::Runtime:

Public Member Functions

- virtual ~**Runtime** ()

Static Public Member Functions

- static **Runtime** * **getRuntime** ()

  *Gets the single instance of the Decaf **Runtime** (p. 2257) for this Process.*

- static void **initializeRuntime** (int argc, char **argv)

  Initialize the Decaf Library passing it the args that were passed to the application at startup.

- static void **initializeRuntime** ()

  Initialize the Decaf Library.

- static void **shutdownRuntime** ()

  Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

6.473.1 Constructor & Destructor Documentation

6.473.1.1 virtual decaf::lang::Runtime::~Runtime () `[inline, virtual]`

6.473.2 Member Function Documentation

6.473.2.1 static Runtime* decaf::lang::Runtime::getRuntime () `[static]`

Gets the single instance of the Decaf **Runtime** (p. 2257) for this Process.

Returns:

pointer to the single Decaf **Runtime** (p. 2257) instance that exists for this process

6.473.2.2 static void decaf::lang::Runtime::initializeRuntime () `[static]`

Initialize the Decaf Library.

Exceptions:

runtime_error if the library is already initialized or an error occurs during initialization.

6.473.2.3 static void decaf::lang::Runtime::initializeRuntime (int *argc*, char ** *argv*) [static]

Initialize the Decaf Library passing it the args that were passed to the application at startup.

Parameters:

argc - The number of args passed

argv - Array of char* values passed to the Process on start.

Exceptions:

runtime_error if the library is already initialized or an error occurs during initialization.

6.473.2.4 static void decaf::lang::Runtime::shutdownRuntime () [static]

Shutdown the Decaf Library, this call should take places after all objects that were created from the Decaf library have been deallocated.

Exceptions:

runtime_error if the library has not already been initialized or an error occurs during shutdown.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Runtime.h**

6.474 decaf::lang::exceptions::RuntimeException Class Reference

`#include <src/main/decaf/lang/exceptions/RuntimeException.h>`Inheritance diagram for decaf::lang::exceptions::RuntimeException:

Public Member Functions

- **RuntimeException** () throw ()
  Default Constructor.
- **RuntimeException** (const **Exception** &ex) throw ()
  Conversion Constructor from some other ActiveMQException.
- **RuntimeException** (const **RuntimeException** &ex) throw ()
  Copy Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **RuntimeException** (const std::exception ***cause**) throw ()
  Constructor.
- **RuntimeException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **RuntimeException** * **clone** () const
  Clones this exception.
- virtual ~**RuntimeException** () throw ()

6.474.1 Constructor & Destructor Documentation

6.474.1.1 decaf::lang::exceptions::RuntimeException::RuntimeException () throw () `[inline]`

Default Constructor.

6.474.1.2 decaf::lang::exceptions::RuntimeException::RuntimeException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other ActiveMQException.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.474.1.3 decaf::lang::exceptions::RuntimeException::RuntimeException (const RuntimeException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The **Exception** (p. 1268) whose data is to be copied into this one.

6.474.1.4 decaf::lang::exceptions::RuntimeException::RuntimeException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.474.1.5 decaf::lang::exceptions::RuntimeException::RuntimeException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.474.1.6 decaf::lang::exceptions::RuntimeException::RuntimeException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.474.1.7 virtual decaf::lang::exceptions::RuntimeException::~RuntimeException () throw () [inline, virtual]

6.474.2 Member Function Documentation

6.474.2.1 virtual RuntimeException* decaf::lang::exceptions::RuntimeException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**RuntimeException.h**

6.475 decaf::security_provider::SecurityProvider Class Reference

`#include <src/main/decaf/security_provider/SecurityProvider.h>`

Public Member Functions

- virtual ~**SecurityProvider** ()
- virtual X500Principal * **createX500Principal** (const std::string &name)=0
- virtual X500Principal * **createX500Principal** (InputStream &is)=0
- virtual X500Principal * **createX500Principal** (unsigned char *buffer, int offset, int len)=0

6.475.1 Constructor & Destructor Documentation

6.475.1.1 virtual decaf::security_provider::SecurityProvider::~SecurityProvider () [inline, virtual]

6.475.2 Member Function Documentation

6.475.2.1 virtual X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (unsigned char * *buffer*, int *offset*, int *len*) [pure virtual]

6.475.2.2 virtual X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (InputStream & *is*) [pure virtual]

6.475.2.3 virtual X500Principal* decaf::security_provider::SecurityProvider::createX500Principal (const std::string & *name*) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/**SecurityProvider.h**

6.476 decaf::security_provider::SecurityProviderMap Class Reference

Lookup Map for Connector Factories.

`#include <src/main/decaf/security_provider/SecurityProviderMap.h>`

Public Member Functions

- void **registerSecurityProvider** (const std::string &name, **SecurityProvider** *provider)

  Registers a new provider with this map.

- void **unregisterSecurityProvider** (const std::string &name)

  Unregisters a provider from this map.

- **SecurityProvider** * **lookup** (const std::string &name)

  Lookup the named provider in the Map.

- std::size_t **getSecurityProviderNames** (std::vector< std::string > &providers)

  Fetch a list of provider names that this Map contains.

Static Public Member Functions

- static **SecurityProviderMap** * **getInstance** ()

  Gets a singleton instance of this class.

6.476.1 Detailed Description

Lookup Map for Connector Factories. Use the Connector name to find the associated factory. This class does not take ownership of the stored factories, they must be deallocated somewhere.

6.476.2 Member Function Documentation

6.476.2.1 static SecurityProviderMap* decaf::security_provider::SecurityProviderMap::getInstance () [static]

Gets a singleton instance of this class.

Referenced by decaf::security_provider::SecurityProviderRegistrar::SecurityProviderRegistrar(), and decaf::security_provider::SecurityProviderRegistrar::~SecurityProviderRegistrar().

6.476.2.2 std::size_t decaf::security_provider::SecurityProviderMap::getSecurityProviderNames (std::vector< std::string > & *providers*)

Fetch a list of provider names that this Map contains.

Parameters:

providers A vector object to receive the list

Returns:

count of providers.

6.476.2.3 SecurityProvider* decaf::security_provider::SecurityProviderMap::lookup (const std::string & *name*)

Lookup the named provider in the Map.

Parameters:

name the provider name to lookup

Returns:

the provider assciated with the name, or NULL

6.476.2.4 void decaf::security_provider::SecurityProviderMap::registerSecurityProvider (const std::string & *name*, SecurityProvider * *provider*)

Registers a new provider with this map.

Parameters:

name A name to associate the provider with

provider the provider object to store in the map.

6.476.2.5 void decaf::security_provider::SecurityProviderMap::unregisterSecurityProvider (const std::string & *name*)

Unregisters a provider from this map.

Parameters:

name the name of the provider to remove

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/**SecurityProviderMap.h**

6.477 decaf::security_provider::SecurityProviderRegistrar Class Reference

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

`#include <src/main/decaf/security_provider/SecurityProviderRegistrar.h>`

Public Member Functions

- **SecurityProviderRegistrar** (const std::string &name, **SecurityProvider** *provider, bool manageLifetime=true)

  Creates a registrar and registers the provider with the provider map.

- virtual ~**SecurityProviderRegistrar** ()

  Unregisters the provider from the provider map and destroys it if `manageLifetime` is set.

- virtual **SecurityProvider** * **getProvider** ()

  get a reference to the factory that this class is holding

6.477.1 Detailed Description

Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

6.477.2 Constructor & Destructor Documentation

6.477.2.1 decaf::security_provider::SecurityProviderRegistrar::SecurityProviderRegistrar (const std::string & *name*, SecurityProvider * *provider*, bool *manageLifetime* = `true`) `[inline]`

Creates a registrar and registers the provider with the provider map.

Parameters:

name name of the provider to register

provider the provider object

manageLifetime boolean indicating if this object manages the lifetime of the factory that is being registered.

References decaf::security_provider::SecurityProviderMap::getInstance().

6.477.2.2 virtual decaf::security_provider::SecurityProviderRegistrar::~SecurityProviderRegistrar () `[inline, virtual]`

Unregisters the provider from the provider map and destroys it if `manageLifetime` is set.

References decaf::security_provider::SecurityProviderMap::getInstance().

6.477.3 Member Function Documentation

6.477.3.1 virtual SecurityProvider* decaf::security_provider::SecurityProviderRegistrar::getProvider () [inline, virtual]

get a reference to the factory that this class is holding

Returns:

reference to a factory class

The documentation for this class was generated from the following file:

- src/main/decaf/security_provider/**SecurityProviderRegistrar.h**

6.478 activemq::cmsutil::CmsTemplate::SendExecutor Class Reference

`#include <src/main/activemq/cmsutil/CmsTemplate.h>`Inheritance diagram for activemq::cmsutil::CmsTemplate::SendExecutor:

Public Member Functions

- **SendExecutor** (**MessageCreator** *messageCreator, **CmsTemplate** *parent)
- virtual ~**SendExecutor** ()
- virtual void **doInCms** (**cms::Session** *session, **cms::MessageProducer** *producer) throw (cms::CMSException)

  Execute an action given a session and producer.

6.478.1 Constructor & Destructor Documentation

6.478.1.1 activemq::cmsutil::CmsTemplate::SendExecutor::SendExecutor (MessageCreator * *messageCreator*, CmsTemplate * *parent*) `[inline]`

6.478.1.2 virtual activemq::cmsutil::CmsTemplate::SendExecutor::~SendExecutor () `[inline, virtual]`

6.478.2 Member Function Documentation

6.478.2.1 virtual void activemq::cmsutil::CmsTemplate::SendExecutor::doInCms (cms::Session * *session*, cms::MessageProducer * *producer*) throw (cms::CMSException) `[inline, virtual]`

Execute an action given a session and producer.

Parameters:

session the CMS `Session`

producer the CMS `Producer`

Exceptions:

cms::CMSException *(*p. ***850)*** if thrown by CMS API methods

Implements **activemq::cmsutil::ProducerCallback** (p. 2102).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**CmsTemplate.h**

6.479 decaf::net::ServerSocket Class Reference

A server socket class (for testing purposes).

`#include <src/main/decaf/net/ServerSocket.h>`

Public Types

- typedef apr_socket_t * **SocketHandle**
- typedef apr_sockaddr_t * **SocketAddress**

Public Member Functions

- **ServerSocket** ()

  Constructor.

- virtual ~**ServerSocket** ()

  Destructor.

- virtual void **bind** (const char *host, int port) throw (SocketException)

  Bind and listen to given IP/dns and port.

- virtual void **bind** (const char *host, int port, int backlog) throw (SocketException)

  Bind and listen to given IP/dns and port.

- virtual **Socket** * **accept** () throw (SocketException)

  Blocks until a client connects to the bound socket.

- virtual void **close** () throw (lang::Exception)

  Closes the server socket.

- virtual bool **isBound** () const

6.479.1 Detailed Description

A server socket class (for testing purposes).

6.479.2 Member Typedef Documentation

6.479.2.1 typedef apr_sockaddr_t* decaf::net::ServerSocket::SocketAddress

6.479.2.2 typedef apr_socket_t* decaf::net::ServerSocket::SocketHandle

6.479.3 Constructor & Destructor Documentation

6.479.3.1 decaf::net::ServerSocket::ServerSocket ()

Constructor. Creates a non-bound server socket.

6.479.3.2 virtual decaf::net::ServerSocket::~ServerSocket () [virtual]

Destructor. Releases socket handle if **close()** (p. 2269) hasn't been called.

6.479.4 Member Function Documentation

6.479.4.1 virtual Socket* decaf::net::ServerSocket::accept () throw (SocketException) [virtual]

Blocks until a client connects to the bound socket.

Returns:

new socket. Never returns NULL.

6.479.4.2 virtual void decaf::net::ServerSocket::bind (const char * *host*, int *port*, int *backlog*) throw (SocketException) [virtual]

Bind and listen to given IP/dns and port.

Parameters:

host IP address or host name.

port TCP port between 1..655535

backlog Size of listen backlog.

6.479.4.3 virtual void decaf::net::ServerSocket::bind (const char * *host*, int *port*) throw (SocketException) [virtual]

Bind and listen to given IP/dns and port.

Parameters:

host IP address or host name.

port TCP port between 1..655535

6.479.4.4 virtual void decaf::net::ServerSocket::close () throw (lang::Exception) [virtual]

Closes the server socket.

6.479.4.5 virtual bool decaf::net::ServerSocket::isBound () const [virtual]

Returns:

true of the server socket is bound.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**ServerSocket.h**

6.480 cms::Session Class Reference

A **Session** (p. 2270) object is a single-threaded context for producing and consuming messages.

`#include <src/main/cms/Session.h>`Inheritance diagram for cms::Session:

Public Types

- enum **AcknowledgeMode** {
  AUTO_ACKNOWLEDGE, **DUPS_OK_ACKNOWLEDGE**, **CLIENT_ACKNOWLEDGE**, **SESSION_TRANSACTED**,
  INDIVIDUAL_ACKNOWLEDGE }

Public Member Functions

- virtual ~**Session** ()
- virtual void **close** ()=0 throw (CMSException)

  Closes this session as well as any active child consumers or producers.

- virtual void **commit** ()=0 throw (CMSException)

  Commits all messages done in this transaction and releases any locks currently held.

- virtual void **rollback** ()=0 throw (CMSException)

  Rolls back all messages done in this transaction and releases any locks currently held.

- virtual void **recover** ()=0 throw (CMSException)

  Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message.

- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination)=0 throw (CMSException)

  *Creates a **MessageConsumer** (p. 1795) for the specified destination.*

- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector)=0 throw (CMSException)

  *Creates a **MessageConsumer** (p. 1795) for the specified destination, using a message selector.*

- virtual **MessageConsumer** * **createConsumer** (const **Destination** *destination, const std::string &selector, bool noLocal)=0 throw (CMSException)

  *Creates a **MessageConsumer** (p. 1795) for the specified destination, using a message selector.*

- virtual **MessageConsumer** * **createDurableConsumer** (const **Topic** *destination, const std::string &name, const std::string &selector, bool noLocal=false)=0 throw (CMSException)

  *Creates a durable subscriber to the specified topic, using a **Message** (p. 1753) selector.*

- virtual **MessageProducer** * **createProducer** (const **Destination** *destination)=0 throw (CMSException)

*Creates a **MessageProducer** (p. 1878) to send messages to the specified destination.*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue)=0 throw (CMSException)

  *Creates a new **QueueBrowser** (p. 2158) to peek at Messages on the given **Queue** (p. 2157).*

- virtual **QueueBrowser** * **createBrowser** (const **cms::Queue** *queue, const std::string &selector)=0 throw (CMSException)

  *Creates a new **QueueBrowser** (p. 2158) to peek at Messages on the given **Queue** (p. 2157).*

- virtual **Queue** * **createQueue** (const std::string &queueName)=0 throw (CMSException)

  *Creates a queue identity given a **Queue** (p. 2157) name.*

- virtual **Topic** * **createTopic** (const std::string &topicName)=0 throw (CMSException)

  *Creates a topic identity given a **Queue** (p. 2157) name.*

- virtual **TemporaryQueue** * **createTemporaryQueue** ()=0 throw (CMSException)

  *Creates a **TemporaryQueue** (p. 2538) object.*

- virtual **TemporaryTopic** * **createTemporaryTopic** ()=0 throw (CMSException)

  *Creates a **TemporaryTopic** (p. 2540) object.*

- virtual **Message** * **createMessage** ()=0 throw (CMSException)

  *Creates a new **Message** (p. 1753).*

- virtual **BytesMessage** * **createBytesMessage** ()=0 throw (CMSException)

  *Creates a **BytesMessage** (p. 759).*

- virtual **BytesMessage** * **createBytesMessage** (const unsigned char *bytes, std::size_t bytesSize)=0 throw (CMSException)

  *Creates a **BytesMessage** (p. 759) and sets the payload to the passed value.*

- virtual **StreamMessage** * **createStreamMessage** ()=0 throw (CMSException)

  *Creates a new **StreamMessage** (p. 2476).*

- virtual **TextMessage** * **createTextMessage** ()=0 throw (CMSException)

  *Creates a new **TextMessage** (p. 2542).*

- virtual **TextMessage** * **createTextMessage** (const std::string &text)=0 throw (CMSException)

  *Creates a new **TextMessage** (p. 2542) and set the text to the value given.*

- virtual **MapMessage** * **createMapMessage** ()=0 throw (CMSException)

  *Creates a new **MapMessage** (p. 1701).*

- virtual **AcknowledgeMode** **getAcknowledgeMode** () const =0 throw (CMSException)

  Returns the acknowledgment mode of the session.

- virtual bool **isTransacted** () const =0 throw (CMSException)

  *Gets if the Sessions is a Transacted **Session** (p. 2270).*

- virtual void **unsubscribe** (const std::string &name)=0 throw (CMSException)

  Unsubscribes a durable subscription that has been created by a client.

6.480.1 Detailed Description

A **Session** (p. 2270) object is a single-threaded context for producing and consuming messages. A session serves several purposes:

- It is a factory for its message producers and consumers.
- It supplies provider-optimized message factories.
- It is a factory for TemporaryTopics and TemporaryQueues.
- It provides a way to create **Queue** (p. 2157) or **Topic** (p. 2571) objects for those clients that need to dynamically manipulate provider-specific destination names.
- It supports a single series of transactions that combine work spanning its producers and consumers into atomic units.
- It defines a serial order for the messages it consumes and the messages it produces.
- It retains messages it consumes until they have been acknowledged.
- It serializes execution of message listeners registered with its message consumers.

A session can create and service multiple message producers and consumers.

One typical use is to have a thread block on a synchronous **MessageConsumer** (p. 1795) until a message arrives. The thread may then use one or more of the Session's MessageProducers.

If a client desires to have one thread produce messages while others consume them, the client should use a separate session for its producing thread.

Certain rules apply to a session's `close` method and are detailed below.

- There is no need to close the producers and consumers of a closed session.
- The close call will block until a receive call or message listener in progress has completed. A blocked message consumer receive call returns null when this session is closed.
- Closing a transacted session must roll back the transaction in progress.
- The close method is the only **Session** (p. 2270) method that can be called concurrently.
- Invoking any other **Session** (p. 2270) method on a closed session must throw an **IllegalStateException** (p. 1399). Closing a closed session must not throw any exceptions.

Transacted Sessions

When a **Session** (p. 2270) is created it can be set to operate in a Transaction based mode. Each **Session** (p. 2270) then operates in a single transaction for all Producers and Consumers of that **Session** (p. 2270). Messages sent and received within a Transaction are grouped into an atomic unit that is committed or rolled back together.

For a **MessageProducer** (p. 1878) this implies that all messages sent by the producer are not sent to the Provider unit the commit call is made. Rolling back the Transaction results in all produced Messages being dropped.

For a **MessageConsumer** (p. 1795) this implies that all received messages are not Acknowledged until the Commit call is made. Rolling back the Transaction results in all Consumed **Message** (p. 1753) being redelivered to the client, the Provider may allow configuration that limits the Maximum number of redeliveries for a **Message** (p. 1753).

Since:

1.0

6.480.2 Member Enumeration Documentation

6.480.2.1 enum cms::Session::AcknowledgeMode

Enumerator:

AUTO_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns.

DUPS_OK_ACKNOWLEDGE With this acknowledgment mode, the session automatically acknowledges a client's receipt of a message either when the session has successfully returned from a call to receive or when the message listener the session has called to process the message successfully returns. Acknowledgments may be delayed in this mode to increase performance at the cost of the message being redelivered this client fails.

CLIENT_ACKNOWLEDGE With this acknowledgment mode, the client acknowledges a consumed message by calling the message's acknowledge method.

SESSION_TRANSACTED Messages will be consumed when the transaction commits.

INDIVIDUAL_ACKNOWLEDGE **Message** (p. 1753) will be acknowledged individually. Normally the acks sent acknowledge the given message and all messages received before it, this mode only acknowledges one message.

6.480.3 Constructor & Destructor Documentation

6.480.3.1 virtual cms::Session::~Session () [inline, virtual]

6.480.4 Member Function Documentation

6.480.4.1 virtual void cms::Session::close () throw (CMSException) [pure virtual]

Closes this session as well as any active child consumers or producers.

Exceptions:

CMSException *(p. 850)* - If an internal error occurs.

Implements **cms::Closeable** (p. 838).

Implemented in **activemq::cmsutil::PooledSession** (p. 2021), and **activemq::core::ActiveMQSession** (p. 356).

6.480.4.2 virtual void cms::Session::commit () throw (CMSException) [pure virtual]

Commits all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

IllegalStateException *(p. **1399**)* - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2021), and **activemq::core::ActiveMQSession** (p. 357).

Referenced by activemq::cmsutil::PooledSession::commit().

6.480.4.3 virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * *queue*, const std::string & *selector*) throw (CMSException) [pure virtual]

Creates a new **QueueBrowser** (p. 2158) to peek at Messages on the given **Queue** (p. 2157).

Parameters:

queue the **Queue** (p. 2157) to browse

selector the **Message** (p. 1753) selector to filter which messages are browsed.

Returns:

New **QueueBrowser** (p. 2158) that is owned by the caller.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

InvalidDestinationException *(p. **1466**)* - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2022), and **activemq::core::ActiveMQSession** (p. 357).

6.480.4.4 virtual QueueBrowser* cms::Session::createBrowser (const cms::Queue * *queue*) throw (CMSException) [pure virtual]

Creates a new **QueueBrowser** (p. 2158) to peek at Messages on the given **Queue** (p. 2157).

Parameters:

queue the **Queue** (p. 2157) to browse

Returns:

New **QueueBrowser** (p. 2158) that is owned by the caller.

Exceptions:

CMSException *(*p. ***850***) - If an internal error occurs.

InvalidDestinationException *(*p. ***1466***) - if the destination given is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2022), and **activemq::core::ActiveMQSession** (p. 357).

6.480.4.5 virtual BytesMessage* cms::Session::createBytesMessage (const unsigned char * *bytes*, std::size_t *bytesSize*) throw (CMSException) [pure virtual]

Creates a **BytesMessage** (p. 759) and sets the payload to the passed value.

Parameters:

bytes an array of bytes to set in the message

bytesSize the size of the bytes array, or number of bytes to use

Exceptions:

CMSException *(*p. ***850***) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2023), and **activemq::core::ActiveMQSession** (p. 358).

6.480.4.6 virtual BytesMessage* cms::Session::createBytesMessage () throw (CMSException) [pure virtual]

Creates a **BytesMessage** (p. 759).

Exceptions:

CMSException *(*p. ***850***) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2023), and **activemq::core::ActiveMQSession** (p. 358).

Referenced by activemq::cmsutil::PooledSession::createBytesMessage().

6.480.4.7 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination*, const std::string & *selector*, bool *noLocal*) throw (CMSException) [pure virtual]

Creates a **MessageConsumer** (p. 1795) for the specified destination, using a message selector.

Parameters:

destination the **Destination** (p. 1190) that this consumer receiving messages for.

selector the **Message** (p. 1753) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new **MessageConsumer** (p. 1795) that is owned by the caller (caller deletes)

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

InvalidDestinationException *(p. **1466**)* - if an invalid destination is specified.

InvalidSelectorException *(p. **1473**)* - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2024), and **activemq::core::ActiveMQSession** (p. 358).

6.480.4.8 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination*, const std::string & *selector*) throw (CMSException) [pure virtual]

Creates a **MessageConsumer** (p. 1795) for the specified destination, using a message selector.

Parameters:

destination the **Destination** (p. 1190) that this consumer receiving messages for.

selector the **Message** (p. 1753) Selector to use

Returns:

pointer to a new **MessageConsumer** (p. 1795) that is owned by the caller (caller deletes)

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

InvalidDestinationException *(p. **1466**)* - if an invalid destination is specified.

InvalidSelectorException *(p. **1473**)* - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2024), and **activemq::core::ActiveMQSession** (p. 359).

6.480.4.9 virtual MessageConsumer* cms::Session::createConsumer (const Destination * *destination*) throw (CMSException) [pure virtual]

Creates a **MessageConsumer** (p. 1795) for the specified destination.

Parameters:

destination the **Destination** (p. 1190) that this consumer receiving messages for.

Returns:

pointer to a new **MessageConsumer** (p. 1795) that is owned by the caller (caller deletes)

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

InvalidDestinationException *(*p. ***1466***)* - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2025), and **activemq::core::ActiveMQSession** (p. 359).

Referenced by activemq::cmsutil::PooledSession::createConsumer().

6.480.4.10 virtual MessageConsumer* cms::Session::createDurableConsumer (const Topic * *destination*, const std::string & *name*, const std::string & *selector*, bool *noLocal* = `false`) throw (CMSException) [pure virtual]

Creates a durable subscriber to the specified topic, using a **Message** (p. 1753) selector. Sessions that create durable consumers must use the same client Id as was used the last time the subscription was created in order to receive all messages that were delivered while the client was offline.

Parameters:

destination the topic to subscribe to

name The name used to identify the subscription

selector the **Message** (p. 1753) Selector to use

noLocal if true, and the destination is a topic, inhibits the delivery of messages published by its own connection. The behavior for NoLocal is not specified if the destination is a queue.

Returns:

pointer to a new durable **MessageConsumer** (p. 1795) that is owned by the caller (caller deletes)

Exceptions:

CMSException *(*p. ***850***)* - If an internal error occurs.

InvalidDestinationException *(*p. ***1466***)* - if an invalid destination is specified.

InvalidSelectorException *(*p. ***1473***)* - if the message selector is invalid.

Implemented in **activemq::cmsutil::PooledSession** (p. 2025), and **activemq::core::ActiveMQSession** (p. 359).

Referenced by activemq::cmsutil::PooledSession::createDurableConsumer().

6.480.4.11 virtual MapMessage* cms::Session::createMapMessage () throw (CMSException) [pure virtual]

Creates a new **MapMessage** (p. 1701).

Exceptions:

CMSException *(*p. ***850***)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2026), and **activemq::core::ActiveMQSession** (p. 360).

Referenced by activemq::cmsutil::PooledSession::createMapMessage().

6.480.4.12 virtual Message* cms::Session::createMessage () throw (CMSException) [pure virtual]

Creates a new **Message** (p. 1753).

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2026), and **activemq::core::ActiveMQSession** (p. 360).

Referenced by activemq::cmsutil::PooledSession::createMessage().

6.480.4.13 virtual MessageProducer* cms::Session::createProducer (const Destination * *destination*) throw (CMSException) [pure virtual]

Creates a **MessageProducer** (p. 1878) to send messages to the specified destination.

Parameters:

destination the **Destination** (p. 1190) to send on

Returns:

New **MessageProducer** (p. 1878) that is owned by the caller.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

InvalidDestinationException *(p. **1466**)* - if an invalid destination is specified.

Implemented in **activemq::cmsutil::PooledSession** (p. 2026), and **activemq::core::ActiveMQSession** (p. 360).

Referenced by activemq::cmsutil::PooledSession::createProducer().

6.480.4.14 virtual Queue* cms::Session::createQueue (const std::string & *queueName*) throw (CMSException) [pure virtual]

Creates a queue identity given a **Queue** (p. 2157) name.

Parameters:

queueName the name of the new **Queue** (p. 2157)

Returns:

new **Queue** (p. 2157) pointer that is owned by the caller.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2027), and **activemq::core::ActiveMQSession** (p. 361).

Referenced by activemq::cmsutil::PooledSession::createQueue().

6.480.4.15 virtual StreamMessage* cms::Session::createStreamMessage () throw (CMSException) [pure virtual]

Creates a new **StreamMessage** (p. 2476).

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2027), and **activemq::core::ActiveMQSession** (p. 361).

Referenced by activemq::cmsutil::PooledSession::createStreamMessage().

6.480.4.16 virtual TemporaryQueue* cms::Session::createTemporaryQueue () throw (CMSException) [pure virtual]

Creates a **TemporaryQueue** (p. 2538) object.

Returns:

new **TemporaryQueue** (p. 2538) pointer that is owned by the caller.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2028), and **activemq::core::ActiveMQSession** (p. 361).

Referenced by activemq::cmsutil::PooledSession::createTemporaryQueue().

6.480.4.17 virtual TemporaryTopic* cms::Session::createTemporaryTopic () throw (CMSException) [pure virtual]

Creates a **TemporaryTopic** (p. 2540) object.

Exceptions:

CMSException *(p. **850**)* - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2028), and **activemq::core::ActiveMQSession** (p. 361).

Referenced by activemq::cmsutil::PooledSession::createTemporaryTopic().

6.480.4.18 virtual TextMessage* cms::Session::createTextMessage (const std::string & *text*) throw (CMSException) [pure virtual]

Creates a new **TextMessage** (p. 2542) and set the text to the value given.

Parameters:

text the initial text for the message

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2028), and **activemq::core::ActiveMQSession** (p. 362).

6.480.4.19 virtual TextMessage* cms::Session::createTextMessage () throw (CMSException) [pure virtual]

Creates a new **TextMessage** (p. 2542).

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2028), and **activemq::core::ActiveMQSession** (p. 362).

Referenced by activemq::cmsutil::PooledSession::createTextMessage().

6.480.4.20 virtual Topic* cms::Session::createTopic (const std::string & *topicName*) throw (CMSException) [pure virtual]

Creates a topic identity given a **Queue** (p. 2157) name.

Parameters:

topicName the name of the new **Topic** (p. 2571)

Returns:

new **Topic** (p. 2571) pointer that is owned by the caller.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2029), and **activemq::core::ActiveMQSession** (p. 362).

Referenced by activemq::cmsutil::PooledSession::createTopic().

6.480.4.21 virtual AcknowledgeMode cms::Session::getAcknowledgeMode () const throw (CMSException) [pure virtual]

Returns the acknowledgment mode of the session.

Returns:

the Sessions Acknowledge Mode

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2029), and **activemq::core::ActiveMQSession** (p. 364).

Referenced by activemq::cmsutil::PooledSession::getAcknowledgeMode().

6.480.4.22 virtual bool cms::Session::isTransacted () const throw (CMSException) [pure virtual]

Gets if the Sessions is a Transacted **Session** (p. 2270).

Returns:

transacted true - false.

Exceptions:

CMSException (p. ***850***) - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2030), and **activemq::core::ActiveMQSession** (p. 365).

Referenced by activemq::cmsutil::PooledSession::isTransacted().

6.480.4.23 virtual void cms::Session::recover () throw (CMSException) [pure virtual]

Stops message delivery in this session, and restarts message delivery with the oldest unacknowledged message. All consumers deliver messages in a serial order. Acknowledging a received message automatically acknowledges all messages that have been delivered to the client.

Restarting a session causes it to take the following actions:

- Stop message delivery
- Mark all messages that might have been delivered but not acknowledged as "redelivered"
- Restart the delivery sequence including all unacknowledged messages that had been previously delivered. Redelivered messages do not have to be delivered in exactly their original delivery order.

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to stop and restart message delivery due to some internal error.

IllegalStateException (p. ***1399***) - if the method is called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2030), and **activemq::core::ActiveMQSession** (p. 366).

Referenced by activemq::cmsutil::PooledSession::recover().

6.480.4.24 virtual void cms::Session::rollback () throw (CMSException) [pure virtual]

Rolls back all messages done in this transaction and releases any locks currently held.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

IllegalStateException *(*p. ***1399)*** - if the method is not called by a transacted session.

Implemented in **activemq::cmsutil::PooledSession** (p. 2031), and **activemq::core::ActiveMQSession** (p. 366).

Referenced by activemq::cmsutil::PooledSession::rollback().

6.480.4.25 virtual void cms::Session::unsubscribe (const std::string & *name*) throw (CMSException) [pure virtual]

Unsubscribes a durable subscription that has been created by a client. This method deletes the state being maintained on behalf of the subscriber by its provider. It is erroneous for a client to delete a durable subscription while there is an active **MessageConsumer** (p. 1795) or Subscriber for the subscription, or while a consumed message is part of a pending transaction or has not been acknowledged in the session.

Parameters:

name The name used to identify this subscription

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::cmsutil::PooledSession** (p. 2031), and **activemq::core::ActiveMQSession** (p. 367).

Referenced by activemq::cmsutil::PooledSession::unsubscribe().

The documentation for this class was generated from the following file:

- src/main/cms/**Session.h**

6.481 activemq::cmsutil::SessionCallback Class Reference

Callback for executing any number of operations on a provided CMS Session.

`#include <src/main/activemq/cmsutil/SessionCallback.h>`Inheritance diagram for activemq::cmsutil::SessionCallback:

Public Member Functions

- virtual ~**SessionCallback** ()
- virtual void **doInCms** (**cms::Session** *session)=0 throw (cms::CMSException)

  Execute any number of operations against the supplied CMS session.

6.481.1 Detailed Description

Callback for executing any number of operations on a provided CMS Session.

6.481.2 Constructor & Destructor Documentation

6.481.2.1 virtual activemq::cmsutil::SessionCallback::~SessionCallback () [inline, virtual]

6.481.3 Member Function Documentation

6.481.3.1 virtual void activemq::cmsutil::SessionCallback::doInCms (cms::Session * *session*) throw (cms::CMSException) [pure virtual]

Execute any number of operations against the supplied CMS session.

Parameters:

session the CMS `Session`

Exceptions:

cms::CMSException *(*p. *850)* if thrown by CMS API methods

Implemented in **activemq::cmsutil::CmsTemplate::ProducerExecutor** (p. 2103), and **activemq::cmsutil::CmsTemplate::ReceiveExecutor** (p. 2172).

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionCallback.h**

6.482 activemq::commands::SessionId Class Reference

`#include <src/main/activemq/commands/SessionId.h>`Inheritance diagram for activemq::commands::SessionId:

Public Types

- typedef **decaf::lang::PointerComparator< SessionId > COMPARATOR**

Public Member Functions

- **SessionId** ()
- **SessionId** (const **SessionId** &other)
- **SessionId** (const **ConnectionId** ∗**connectionId**, long long sessionId)
- **SessionId** (const **ProducerId** ∗producerId)
- **SessionId** (const **ConsumerId** ∗consumerId)
- virtual ∼**SessionId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **SessionId** ∗ **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** ∗src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** ∗**value**) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- const **Pointer< ConnectionId >** & **getParentId** () const
- virtual const std::string & **getConnectionId** () const
- virtual std::string & **getConnectionId** ()
- virtual void **setConnectionId** (const std::string &**connectionId**)
- virtual long long **getValue** () const
- virtual void **setValue** (long long **value**)
- virtual int **compareTo** (const **SessionId** &**value**) const
- virtual bool **equals** (const **SessionId** &**value**) const
- virtual bool **operator==** (const **SessionId** &**value**) const
- virtual bool **operator<** (const **SessionId** &**value**) const
- **SessionId** & **operator=** (const **SessionId** &other)

Static Public Attributes

- static const unsigned char **ID_SESSIONID** = 121

Protected Attributes

- std::string **connectionId**
- long long **value**

6.482.1 Member Typedef Documentation

6.482.1.1 typedef decaf::lang::PointerComparator<SessionId> activemq::commands::SessionId::COMPARATOR

6.482.2 Constructor & Destructor Documentation

6.482.2.1 activemq::commands::SessionId::SessionId ()

6.482.2.2 activemq::commands::SessionId::SessionId (const SessionId & *other*)

6.482.2.3 activemq::commands::SessionId::SessionId (const ConnectionId * *connectionId*, long long *sessionId*)

6.482.2.4 activemq::commands::SessionId::SessionId (const ProducerId * *producerId*)

6.482.2.5 activemq::commands::SessionId::SessionId (const ConsumerId * *consumerId*)

6.482.2.6 virtual activemq::commands::SessionId::~SessionId () [virtual]

6.482.3 Member Function Documentation

6.482.3.1 virtual SessionId* activemq::commands::SessionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.482.3.2 virtual int activemq::commands::SessionId::compareTo (const SessionId & *value*) const [virtual]

6.482.3.3 virtual void activemq::commands::SessionId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.482.3.4 virtual bool activemq::commands::SessionId::equals (const SessionId & *value*) const [virtual]

6.482.3.5 virtual bool activemq::commands::SessionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.482.3.6 virtual std::string& activemq::commands::SessionId::getConnectionId () [virtual]

6.482.3.7 virtual const std::string& activemq::commands::SessionId::getConnectionId () const [virtual]

Referenced by activemq::commands::ConsumerId::ConsumerId(), and activemq::commands::ProducerId::ProducerId().

6.482.3.8 virtual unsigned char activemq::commands::SessionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.482.3.9 const Pointer<ConnectionId>& activemq::commands::SessionId::getParentId () const

6.482.3.10 virtual long long activemq::commands::SessionId::getValue () const [virtual]

Referenced by activemq::commands::ConsumerId::ConsumerId(), and activemq::commands::ProducerId::ProducerId().

6.482.3.11 virtual bool activemq::commands::SessionId::operator< (const SessionId & *value*) const [virtual]

6.482.3.12 SessionId& activemq::commands::SessionId::operator= (const SessionId & *other*)

6.482.3.13 virtual bool activemq::commands::SessionId::operator== (const SessionId & *value*) const [virtual]

6.482.3.14 virtual void activemq::commands::SessionId::setConnectionId (const std::string & *connectionId*) [virtual]

6.482.3.15 virtual void activemq::commands::SessionId::setValue (long long *value*) [virtual]

6.482.3.16 virtual std::string activemq::commands::SessionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.482.4 Field Documentation

6.482.4.1 std::string activemq::commands::SessionId::connectionId [protected]

6.482.4.2 const unsigned char activemq::commands::SessionId::ID_SESSIONID = 121 [static]

Referenced by activemq::state::CommandVisitorAdapter::processRemoveInfo().

6.482.4.3 long long activemq::commands::SessionId::value [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SessionId.h**

6.483 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2288).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.483.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2288). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.483.2 Constructor & Destructor Documentation

6.483.2.1 activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::SessionIdMarshaller () [inline]

6.483.2.2 virtual activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::~SessionIdMarshaller () [inline, virtual]

6.483.3 Member Function Documentation

6.483.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.483.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.483.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.483.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.483.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.483.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.483.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionIdMarshaller.h**

6.484 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2292).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.484.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2292). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.484.2 Constructor & Destructor Documentation

6.484.2.1 activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::SessionIdMarshaller () [inline]

6.484.2.2 virtual activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::~SessionIdMarshaller () [inline, virtual]

6.484.3 Member Function Documentation

6.484.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.484.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.484.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.484.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.484.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.484.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.484.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SessionIdMarshaller.h**

6.485 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller Class Reference

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2296).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller:

Public Member Functions

- **SessionIdMarshaller** ()
- virtual ~**SessionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.485.1 Detailed Description

Marshaling code for Open Wire Format for **SessionIdMarshaller** (p. 2296). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.485.2 Constructor & Destructor Documentation

6.485.2.1 activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::SessionIdMarshaller () [inline]

6.485.2.2 virtual activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::~SessionIdMarshaller () [inline, virtual]

6.485.3 Member Function Documentation

6.485.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.485.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.485.3.3 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.485.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.485.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.485.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.485.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SessionIdMarshaller.h**

6.486 activemq::commands::SessionInfo Class Reference

`#include <src/main/activemq/commands/SessionInfo.h>`Inheritance diagram for activemq::commands::SessionInfo:

Public Member Functions

- **SessionInfo** ()
- virtual ~**SessionInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **SessionInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- unsigned int **getAckMode** () const
- void **setAckMode** (unsigned int mode)
- virtual const **Pointer**< **SessionId** > & **getSessionId** () const
- virtual **Pointer**< **SessionId** > & **getSessionId** ()
- virtual void **setSessionId** (const **Pointer**< **SessionId** > &**sessionId**)
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SESSIONINFO** = 4

Protected Member Functions

- **SessionInfo** (const **SessionInfo** &)
- **SessionInfo** & **operator**= (const **SessionInfo** &)

Protected Attributes

- **Pointer< SessionId > sessionId**

6.486.1 Constructor & Destructor Documentation

6.486.1.1 activemq::commands::SessionInfo::SessionInfo (const SessionInfo &) [inline, protected]

6.486.1.2 activemq::commands::SessionInfo::SessionInfo ()

6.486.1.3 virtual activemq::commands::SessionInfo::~SessionInfo () [virtual]

6.486.2 Member Function Documentation

6.486.2.1 virtual SessionInfo* activemq::commands::SessionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.486.2.2 virtual void activemq::commands::SessionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.486.2.3 virtual bool activemq::commands::SessionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.486.2.4 unsigned int activemq::commands::SessionInfo::getAckMode () const [inline]

6.486.2.5 virtual unsigned char activemq::commands::SessionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.486.2.6 virtual Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () [virtual]

6.486.2.7 virtual const Pointer<SessionId>& activemq::commands::SessionInfo::getSessionId () const [virtual]

6.486.2.8 SessionInfo& activemq::commands::SessionInfo::operator= (const SessionInfo &) [inline, protected]

6.486.2.9 void activemq::commands::SessionInfo::setAckMode (unsigned int *mode*) [inline]

6.486.2.10 virtual void activemq::commands::SessionInfo::setSessionId (const Pointer< SessionId > & *sessionId*) [virtual]

6.486.2.11 virtual std::string activemq::commands::SessionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.486.2.12 virtual Pointer<Command> activemq::commands::SessionInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.486.3 Field Documentation

6.486.3.1 const unsigned char activemq::commands::SessionInfo::ID_SESSIONINFO = 4 [static]

6.486.3.2 Pointer<SessionId> activemq::commands::SessionInfo::sessionId [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SessionInfo.h**

6.487 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshall Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2304).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/SessionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual ~**SessionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.487.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2304). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.487.2 Constructor & Destructor Documentation

6.487.2.1 activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::SessionInfoMarshalle () [inline]

6.487.2.2 virtual activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::~SessionInfoMarshall () [inline, virtual]

6.487.3 Member Function Documentation

6.487.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.487.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.487.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.487.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.487.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.487.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.487.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SessionInfoMarshaller.h**

6.488 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshall Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2308).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/SessionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual ~**SessionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.488.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2308). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.488.2 Constructor & Destructor Documentation

6.488.2.1 activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::SessionInfoMarshalle () [inline]

6.488.2.2 virtual activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::~SessionInfoMarshall () [inline, virtual]

6.488.3 Member Function Documentation

6.488.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.488.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.488.3.3 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.488.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.488.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.488.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.488.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SessionInfoMarshaller.h**

6.489 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshalle Class Reference

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2312).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/SessionInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller:

Public Member Functions

- **SessionInfoMarshaller** ()
- virtual ~**SessionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.489.1 Detailed Description

Marshaling code for Open Wire Format for **SessionInfoMarshaller** (p. 2312). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.489.2 Constructor & Destructor Documentation

6.489.2.1 activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::SessionInfoMarshalle () [inline]

6.489.2.2 virtual activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::~SessionInfoMarshall () [inline, virtual]

6.489.3 Member Function Documentation

6.489.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.489.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::getDataStructureType () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.489.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.489.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.489.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.489.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.489.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SessionInfoMarshaller.h**

6.490 activemq::cmsutil::SessionPool Class Reference

A pool of CMS sessions from the same connection and with the same acknowledge mode.

`#include <src/main/activemq/cmsutil/SessionPool.h>`

Public Member Functions

- **SessionPool** (**cms::Connection** *connection, **cms::Session::AcknowledgeMode** ackMode, **ResourceLifecycleManager** *resourceLifecycleManager)

  Constructs a session pool.

- virtual ~**SessionPool** ()

  Destroys the pooled session objects, but not the underlying session resources.

- virtual **PooledSession** * **takeSession** () throw (cms::CMSException)

  Takes a session from the pool, creating one if necessary.

- virtual void **returnSession** (**PooledSession** *session)

  Returns a session to the pool.

- **ResourceLifecycleManager** * **getResourceLifecycleManager** ()

6.490.1 Detailed Description

A pool of CMS sessions from the same connection and with the same acknowledge mode. Internal session resources are managed through a provided `ResourceLifecycleManager (p. 2228)`, not by this pool. This class is thread-safe.

6.490.2 Constructor & Destructor Documentation

6.490.2.1 activemq::cmsutil::SessionPool::SessionPool (cms::Connection * *connection*, cms::Session::AcknowledgeMode *ackMode*, ResourceLifecycleManager * *resourceLifecycleManager*)

Constructs a session pool.

Parameters:

connection the connection to be used for creating all sessions.

ackMode the acknowledge mode to be used for all sessions

resourceLifecycleManager the object responsible for managing the lifecycle of any allocated **cms::Session** (p. 2270) resources.

6.490.2.2 virtual activemq::cmsutil::SessionPool::~SessionPool () `[virtual]`

Destroys the pooled session objects, but not the underlying session resources. That is the job of the **ResourceLifecycleManager** (p. 2228).

6.490.3 Member Function Documentation

6.490.3.1 ResourceLifecycleManager* activemq::cmsutil::SessionPool::getResourceLifecycleManager () [inline]

6.490.3.2 virtual void activemq::cmsutil::SessionPool::returnSession (PooledSession * *session*) [virtual]

Returns a session to the pool.

Parameters:

session the session to be returned.

6.490.3.3 virtual PooledSession* activemq::cmsutil::SessionPool::takeSession () throw (cms::CMSException) [virtual]

Takes a session from the pool, creating one if necessary.

Returns:

the pooled session object

Exceptions:

cms::CMSException (p. ***850***) if an error occurred

The documentation for this class was generated from the following file:

- src/main/activemq/cmsutil/**SessionPool.h**

6.491 activemq::state::SessionState Class Reference

`#include <src/main/activemq/state/SessionState.h>`

Public Member Functions

- **SessionState** (const **Pointer**< **SessionInfo** > &info)
- virtual ~**SessionState** ()
- std::string **toString** () const
- const **Pointer**< **SessionInfo** > **getInfo** () const
- void **addProducer** (const **Pointer**< **ProducerInfo** > &info)
- **Pointer**< **ProducerState** > **removeProducer** (const **Pointer**< **ProducerId** > &id)
- void **addConsumer** (const **Pointer**< **ConsumerInfo** > &info)
- **Pointer**< **ConsumerState** > **removeConsumer** (const **Pointer**< **ConsumerId** > &id)
- std::vector< **Pointer**< **ProducerState** > > **getProducerStates** () const
- **Pointer**< **ProducerState** > **getProducerState** (const **Pointer**< **ProducerId** > &id)
- std::vector< **Pointer**< **ConsumerState** > > **getConsumerStates** () const
- **Pointer**< **ConsumerState** > **getConsumerState** (const **Pointer**< **ConsumerId** > &id)
- void **checkShutdown** () const
- void **shutdown** ()

6.491.1 Constructor & Destructor Documentation

6.491.1.1 activemq::state::SessionState::SessionState (const Pointer< SessionInfo > & *info*)

6.491.1.2 virtual activemq::state::SessionState::~SessionState () [virtual]

6.491.2 Member Function Documentation

6.491.2.1 void activemq::state::SessionState::addConsumer (const Pointer< ConsumerInfo > & *info*) [inline]

6.491.2.2 void activemq::state::SessionState::addProducer (const Pointer< ProducerInfo > & *info*) [inline]

6.491.2.3 void activemq::state::SessionState::checkShutdown () const

6.491.2.4 Pointer<ConsumerState> activemq::state::SessionState::getConsumerState (const Pointer< ConsumerId > & *id*) [inline]

6.491.2.5 std::vector< Pointer<ConsumerState> > activemq::state::SessionState::getConsumerStates () const [inline]

6.491.2.6 const Pointer<SessionInfo> activemq::state::SessionState::getInfo () const [inline]

6.491.2.7 Pointer<ProducerState> activemq::state::SessionState::getProducerState (const Pointer< ProducerId > & *id*) [inline]

6.491.2.8 std::vector< Pointer<ProducerState> > activemq::state::SessionState::getProducerStates () const [inline]

6.491.2.9 Pointer<ConsumerState> activemq::state::SessionState::removeConsumer (const Pointer< ConsumerId > & *id*) [inline]

6.491.2.10 Pointer<ProducerState> activemq::state::SessionState::removeProducer (const Pointer< ProducerId > & *id*) [inline]

6.491.2.11 void activemq::state::SessionState::shutdown () [inline]

6.491.2.12 std::string activemq::state::SessionState::toString () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**SessionState.h**

6.492 decaf::util::Set< E > Class Template Reference

A collection that contains no duplicate elements.

`#include <src/main/decaf/util/Set.h>`Inheritance diagram for decaf::util::Set< E >:

Public Member Functions

- virtual ~**Set** ()

6.492.1 Detailed Description

template<typename E> class decaf::util::Set< E >

A collection that contains no duplicate elements. More formally, sets contain no pair of elements e1 and e2 such that e1 == e2, and at most one null element. As implied by its name, this interface models the mathematical set abstraction.

The additional stipulation on constructors is, not surprisingly, that all constructors must create a set that contains no duplicate elements (as defined above).

Note: Great care must be exercised if mutable objects are used as set elements. The behavior of a set is not specified if the value of an object is changed in a manner that affects equals comparisons while the object is an element in the set.

Since:

1.0

6.492.2 Constructor & Destructor Documentation

6.492.2.1 template<typename E> virtual decaf::util::Set< E >::~Set () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/util/**Set.h**

6.493 decaf::lang::Short Class Reference

`#include <src/main/decaf/lang/Short.h>`Inheritance diagram for decaf::lang::Short:

Public Member Functions

- **Short** (short value)
- **Short** (const std::string &value) throw (exceptions::NumberFormatException)
- virtual ~**Short** ()
- virtual int **compareTo** (const **Short** &s) const

  *Compares this **Short** (p. 2321) instance with another.*

- bool **equals** (const **Short** &s) const
- virtual bool **operator==** (const **Short** &s) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **Short** &s) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual int **compareTo** (const short &s) const

  *Compares this **Short** (p. 2321) instance with another.*

- bool **equals** (const short &s) const
- virtual bool **operator==** (const short &s) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const short &s) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **toString** () const
- virtual double **doubleValue** () const

  Answers the double value which the receiver represents.

- virtual float **floatValue** () const

  Answers the float value which the receiver represents.

- virtual unsigned char **byteValue** () const

  Answers the byte value which the receiver represents.

- virtual short **shortValue** () const

  Answers the short value which the receiver represents.

- virtual int **intValue** () const

  Answers the int value which the receiver represents.

- virtual long long **longValue** () const

  Answers the long value which the receiver represents.

Static Public Member Functions

- static std::string **toString** (short value)
- static **Short decode** (const std::string &value) throw (exceptions::NumberFormatException)

  *Decodes a String into a **Short** (p. 2321).*

- static short **reverseBytes** (short value)

  Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

- static short **parseShort** (const std::string &s, int radix) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed short in the radix specified by the second argument.

- static short **parseShort** (const std::string &s) throw (exceptions::NumberFormatException)

  Parses the string argument as a signed decimal short.

- static **Short valueOf** (short value)

  *Returns a **Short** (p. 2321) instance representing the specified short value.*

- static **Short valueOf** (const std::string &value) throw (exceptions::NumberFormatException)

  *Returns a **Short** (p. 2321) object holding the value given by the specified std::string.*

- static **Short valueOf** (const std::string &value, int radix) throw (exceptions::NumberFormatException)

  *Returns a **Short** (p. 2321) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument.*

Static Public Attributes

- static const int **SIZE** = 16

  Size of this objects primitive type in bits.

- static const short **MAX_VALUE** = (short)0x7FFF

  Max Value for this Object's primitive type.

- static const short **MIN_VALUE** = (short)0x8000

  Max Value for this Object's primitive type.

6.493.1 Constructor & Destructor Documentation

6.493.1.1 decaf::lang::Short::Short (short *value*)

Parameters:

value - short to wrap

6.493.1.2 decaf::lang::Short::Short (const std::string & *value*) throw (exceptions::NumberFormatException)

Parameters:

value - string value to convert to short and wrap

Exceptions:

NumberFormatException

6.493.1.3 virtual decaf::lang::Short::~Short () [inline, virtual]

6.493.2 Member Function Documentation

6.493.2.1 virtual unsigned char decaf::lang::Short::byteValue () const [inline, virtual]

Answers the byte value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1954).

6.493.2.2 virtual int decaf::lang::Short::compareTo (const short & *s*) const [virtual]

Compares this **Short** (p. 2321) instance with another.

Parameters:

s - the **Short** (p. 2321) instance to be compared

Returns:

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

Implements **decaf::lang::Comparable< short >** (p. 899).

6.493.2.3 virtual int decaf::lang::Short::compareTo (const Short & *s*) const [virtual]

Compares this **Short** (p. 2321) instance with another.

Parameters:

s - the **Short** (p. 2321) instance to be compared

Returns:

zero if this object represents the same short value as the argument; a positive value if this object represents a value greater than the passed in value, and -1 if this object repesents a value less than the passed in value.

6.493.2.4 static Short decaf::lang::Short::decode (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Decodes a String into a **Short** (p. 2321). Accepts decimal, hexadecimal, and octal numbers given by the following grammar:

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the **Short.parseShort** (p. 2327) method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a NumberFormatException will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the string.

Parameters:

value - The string to decode

Returns:

a **Short** (p. 2321) object containing the decoded value

Exceptions:

NumberFomatException if the string is not formatted correctly.

6.493.2.5 virtual double decaf::lang::Short::doubleValue () const [inline, virtual]

Answers the double value which the receiver represents.

Returns:

double the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.493.2.6 bool decaf::lang::Short::equals (const short & *s*) const [inline, virtual]

Returns:

true if the two **Short** (p. 2321) Objects have the same value.

Implements **decaf::lang::Comparable< short >** (p. 900).

6.493.2.7 bool decaf::lang::Short::equals (const Short & *s*) const [inline]

Returns:

true if the two **Short** (p. 2321) Objects have the same value.

6.493.2.8 virtual float decaf::lang::Short::floatValue () const [inline, virtual]

Answers the float value which the receiver represents.

Returns:

float the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.493.2.9 virtual int decaf::lang::Short::intValue () const [inline, virtual]

Answers the int value which the receiver represents.

Returns:

int the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.493.2.10 virtual long long decaf::lang::Short::longValue () const [inline, virtual]

Answers the long value which the receiver represents.

Returns:

long the value of the receiver.

Implements **decaf::lang::Number** (p. 1955).

6.493.2.11 virtual bool decaf::lang::Short::operator< (const short & *s*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< short >** (p. 900).

6.493.2.12 virtual bool decaf::lang::Short::operator< (const Short & *s*) const [inline, virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.493.2.13 virtual bool decaf::lang::Short::operator== (const short & *s*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

Implements **decaf::lang::Comparable< short >** (p. 901).

6.493.2.14 virtual bool decaf::lang::Short::operator== (const Short & *s*) const [inline, virtual]

Compares equality between this object and the one passed.

Parameters:

s - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.493.2.15 static short decaf::lang::Short::parseShort (const std::string & *s*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed decimal short. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting short value is returned, exactly as if the argument and the radix 10 were given as arguments to the parseShort(const std::string, int) method.

Parameters:

s - String to convert to a short

Returns:

the converted short value

Exceptions:

NumberFormatException if the string is not a short.

6.493.2.16 static short decaf::lang::Short::parseShort (const std::string & *s*, int *radix*) throw (exceptions::NumberFormatException) [static]

Parses the string argument as a signed short in the radix specified by the second argument. The characters in the string must all be digits, of the specified radix (as determined by whether **Character.digit(char, int)** (p. 804) returns a nonnegative value) except that the first character may be an ASCII minus sign '-' to indicate a negative value. The resulting byte value is returned.

An exception of type NumberFormatException is thrown if any of the following situations occurs: * The first argument is null or is a string of length zero. * The radix is either smaller than **Character.MIN_RADIX** (p. 808) or larger than **Character.MAX_RADIX** (p. 807). * Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' provided that the string is longer than length 1. * The value represented by the string is not a value of type short.

Parameters:

s - the String containing the short representation to be parsed

radix - the radix to be used while parsing s

Returns:

the short represented by the string argument in the specified radix.

Exceptions:

NumberFormatException - If String does not contain a parsable short.

6.493.2.17 static short decaf::lang::Short::reverseBytes (short *value*) [static]

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified short value.

Parameters:

value - the short whose bytes we are to reverse

Returns:

the reversed short.

6.493.2.18 virtual short decaf::lang::Short::shortValue () const [inline, virtual]

Answers the short value which the receiver represents.

Returns:

int the value of the receiver.

Reimplemented from **decaf::lang::Number** (p. 1956).

6.493.2.19 static std::string decaf::lang::Short::toString (short *value*) [static]

Returns:

a string representing the primitive value as Base 10

6.493.2.20 std::string decaf::lang::Short::toString () const

Returns:

this **Short** (p. 2321) Object as a String Representation

6.493.2.21 static Short decaf::lang::Short::valueOf (const std::string & *value*, int *radix*) throw (exceptions::NumberFormatException) [static]

Returns a **Short** (p. 2321) object holding the value extracted from the specified std::string when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed short in the radix specified by the second argument, exactly as if the argument were given to the parseShort(std::string, int) method. The result is a **Short** (p. 2321) object that represents the short value specified by the string.

Parameters:

value - std::string to parse as base (radix)

radix - base of the string to parse.

Returns:

new **Short** (p. 2321) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a valid short.

6.493.2.22 static Short decaf::lang::Short::valueOf (const std::string & *value*) throw (exceptions::NumberFormatException) [static]

Returns a **Short** (p. 2321) object holding the value given by the specified std::string. The argument is interpreted as representing a signed decimal short, exactly as if the argument were given to the parseShort(std::string) method. The result is a **Short** (p. 2321) object that represents the short value specified by the string.

Parameters:

value - std::string to parse as base 10

Returns:

new **Short** (p. 2321) Object wrapping the primitive

Exceptions:

NumberFormatException if the string is not a decimal short.

6.493.2.23 static Short decaf::lang::Short::valueOf (short *value*) [static]

Returns a **Short** (p. 2321) instance representing the specified short value.

Parameters:

value - the short to wrap

Returns:

the new **Short** (p. 2321) object wrapping value.

6.493.3 Field Documentation

6.493.3.1 const short decaf::lang::Short::MAX_VALUE = (short)0x7FFF [static]

Max Value for this Object's primitive type.

6.493.3.2 const short decaf::lang::Short::MIN_VALUE = (short)0x8000 [static]

Max Value for this Object's primitive type.

6.493.3.3 const int decaf::lang::Short::SIZE = 16 [static]

Size of this objects primitive type in bits.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Short.h**

6.494 decaf::internal::nio::ShortArrayBuffer Class Reference

`#include <src/main/decaf/internal/nio/ShortArrayBuffer.h>`Inheritance diagram for decaf::internal::nio::ShortArrayBuffer:

Public Member Functions

- **ShortArrayBuffer** (std::size_t capacity, bool readOnly=false)

  *Creates a **ShortArrayBuffer** (p. 2330) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

- **ShortArrayBuffer** (short *array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::NullPointerException)

  *Creates a **ShortArrayBuffer** (p. 2330) object that wraps the given array.*

- **ShortArrayBuffer** (**ByteArrayPerspective** &array, std::size_t offset, std::size_t capacity, bool readOnly=false) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  *Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset.*

- **ShortArrayBuffer** (const **ShortArrayBuffer** &other)

  *Create a **ShortArrayBuffer** (p. 2330) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.*

- virtual ~**ShortArrayBuffer** ()
- virtual short * **array** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the short array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual ShortBuffer * **asReadOnlyBuffer** () const

  Creates a new, read-only short buffer that shares this buffer's content.

- virtual ShortBuffer & **compact** () throw (decaf::nio::ReadOnlyBufferException)

  Compacts this buffer.

- virtual ShortBuffer * **duplicate** ()

  Creates a new short buffer that shares this buffer's content.

- virtual short **get** () throw (decaf::nio::BufferUnderflowException)

  Relative get method.

- virtual short **get** (std::size_t index) const throw (lang::exceptions::IndexOutOfBoundsException)

Absolute get method.

- virtual bool **hasArray** () const

  Tells whether or not this buffer is backed by an accessible short array.

- virtual bool **isReadOnly** () const

  Tells whether or not this buffer is read-only.

- virtual ShortBuffer & **put** (short value) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the current position, and then increments the position.

- virtual ShortBuffer & **put** (std::size_t index, short value) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException)

  Writes the given doubles into this buffer at the given index.

- virtual ShortBuffer * **slice** () const

  Creates a new ShortBuffer whose content is a shared subsequence of this buffer's content.

Protected Member Functions

- virtual void **setReadOnly** (bool value)

  *Sets this **ByteArrayBuffer** (p. 694) as Read-Only.*

6.494.1 Constructor & Destructor Documentation

6.494.1.1 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (std::size_t *capacity*, bool *readOnly* = `false`)

Creates a **ShortArrayBuffer** (p. 2330) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

6.494.1.2 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (short * *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::NullPointerException)

Creates a **ShortArrayBuffer** (p. 2330) object that wraps the given array. If the own flag is set then it will delete this array when this object is deleted.

Parameters:

array - array to wrap

offset - the position that is this buffers start pos.

capacity - size of the array, this is the limit we read and write to.

readOnly - should this buffer be read-only, default as false

Exceptions:

NullPointerException if buffer is NULL

6.494.1.3 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (ByteArrayPerspective & *array*, std::size_t *offset*, std::size_t *capacity*, bool *readOnly* = `false`) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

Creates a byte buffer that wraps the passed **ByteArrayPerspective** (p. 729) and start at the given offset. The capacity and limit of the new **ShortArrayBuffer** (p. 2330) will be that of the remaining capacity of the passed buffer.

Parameters:

array - the **ByteArrayPerspective** (p. 729) to wrap

offset - the position that is this buffers start pos.

capacity - the length of the array we are wrapping or limit.

readOnly - is this a readOnly buffer.

Exceptions:

IndexOutOfBoundsException if offset is greater than array capacity.

6.494.1.4 decaf::internal::nio::ShortArrayBuffer::ShortArrayBuffer (const ShortArrayBuffer & *other*)

Create a **ShortArrayBuffer** (p. 2330) that mirrors this one, meaning it shares a reference to this buffers **ByteArrayPerspective** (p. 729) and when changes are made to that data it is reflected in both.

Parameters:

other - the **ShortArrayBuffer** (p. 2330) this one is to mirror.

6.494.1.5 virtual decaf::internal::nio::ShortArrayBuffer::~ShortArrayBuffer () `[virtual]`

6.494.2 Member Function Documentation

6.494.2.1 virtual short* decaf::internal::nio::ShortArrayBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) `[virtual]`

Returns the short array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this Buffer

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2341).

6.494.2.2 virtual std::size_t decaf::internal::nio::ShortArrayBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::nio::ReadOnlyBufferException) [virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException if this Buffer is read only.

UnsupportedOperationException if the underlying store has no array.

Implements **decaf::nio::ShortBuffer** (p. 2341).

6.494.2.3 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::asReadOnlyBuffer () const [virtual]

Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only short buffer which the caller then owns.

Implements **decaf::nio::ShortBuffer** (p. 2341).

6.494.2.4 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::compact () throw (decaf::nio::ReadOnlyBufferException) [virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied

to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this ShortBuffer

Exceptions:

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2342).

6.494.2.5 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::duplicate () [virtual]

Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new short Buffer which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2342).

6.494.2.6 virtual short decaf::internal::nio::ShortArrayBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the Buffer where the short is to be read

Returns:

the short that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implements **decaf::nio::ShortBuffer** (p. 2344).

6.494.2.7 virtual short decaf::internal::nio::ShortArrayBuffer::get () throw (decaf::nio::BufferUnderflowException) [virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the short at the current position

Exceptions:

BufferUnderflowException if there no more data to return

Implements **decaf::nio::ShortBuffer** (p. 2344).

6.494.2.8 virtual bool decaf::internal::nio::ShortArrayBuffer::hasArray () const [inline, virtual]

Tells whether or not this buffer is backed by an accessible short array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implements **decaf::nio::ShortBuffer** (p. 2344).

6.494.2.9 virtual bool decaf::internal::nio::ShortArrayBuffer::isReadOnly () const [inline, virtual]

Tells whether or not this buffer is read-only.

Returns:

true if, and only if, this buffer is read-only

Implements **decaf::nio::Buffer** (p. 630).

6.494.2.10 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the given index.

Parameters:

index - position in the Buffer to write the data

value - the doubles to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2345).

6.494.2.11 virtual ShortBuffer& decaf::internal::nio::ShortArrayBuffer::put (short *value*) throw (decaf::nio::BufferOverflowException, decaf::nio::ReadOnlyBufferException) [virtual]

Writes the given doubles into this buffer at the current position, and then increments the position.

Parameters:

value - the doubles value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException - If this buffer is read-only

Implements **decaf::nio::ShortBuffer** (p. 2345).

6.494.2.12 virtual void decaf::internal::nio::ShortArrayBuffer::setReadOnly (bool *value*) [inline, protected, virtual]

Sets this **ByteArrayBuffer** (p. 694) as Read-Only.

Parameters:

value - true if this buffer is to be read-only.

6.494.2.13 virtual ShortBuffer* decaf::internal::nio::ShortArrayBuffer::slice () const [virtual]

Creates a new ShortBuffer whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create ShortBuffer which the caller owns.

Implements **decaf::nio::ShortBuffer** (p. 2347).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/nio/**ShortArrayBuffer.h**

6.495 decaf::nio::ShortBuffer Class Reference

This class defines four categories of operations upon short buffers:.

`#include <src/main/decaf/nio/ShortBuffer.h>`Inheritance diagram for decaf::nio::ShortBuffer:

Public Member Functions

- virtual ~**ShortBuffer** ()
- virtual std::string **toString** () const
- virtual short * **array** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the short array that backs this buffer (optional operation).

- virtual std::size_t **arrayOffset** ()=0 throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException)

  Returns the offset within this buffer's backing array of the first element of the buffer (optional operation).

- virtual **ShortBuffer** * **asReadOnlyBuffer** () const =0

  Creates a new, read-only short buffer that shares this buffer's content.

- virtual **ShortBuffer** & **compact** ()=0 throw (ReadOnlyBufferException)

  Compacts this buffer.

- virtual **ShortBuffer** * **duplicate** ()=0

  Creates a new short buffer that shares this buffer's content.

- virtual short **get** ()=0 throw (BufferUnderflowException)

  Relative get method.

- virtual short **get** (std::size_t index) const =0 throw (lang::exceptions::IndexOutOfBoundsException)

  Absolute get method.

- **ShortBuffer** & **get** (std::vector< short > buffer) throw (BufferUnderflowException)

  Relative bulk get method.

- **ShortBuffer** & **get** (short *buffer, std::size_t offset, std::size_t length) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

  Relative bulk get method.

- virtual bool **hasArray** () const =0

  Tells whether or not this buffer is backed by an accessible short array.

- **ShortBuffer** & **put** (**ShortBuffer** &src) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

  This method transfers the shorts remaining in the given source buffer into this buffer.

- **ShortBuffer** & **put** (const short *buffer, std::size_t offset, std::size_t length) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

  This method transfers shorts into this buffer from the given source array.

- **ShortBuffer** & **put** (std::vector< short > &buffer) throw (BufferOverflowException, ReadOnlyBufferException)

  This method transfers the entire content of the given source shorts array into this buffer.

- virtual **ShortBuffer** & **put** (short value)=0 throw (BufferOverflowException, ReadOnlyBufferException)

  Writes the given shorts into this buffer at the current position, and then increments the position.

- virtual **ShortBuffer** & **put** (std::size_t index, short value)=0 throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException)

  Writes the given shorts into this buffer at the given index.

- virtual **ShortBuffer** * **slice** () const =0

  *Creates a new **ShortBuffer** (p. 2338) whose content is a shared subsequence of this buffer's content.*

- virtual int **compareTo** (const **ShortBuffer** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **ShortBuffer** &value) const
- virtual bool **operator==** (const **ShortBuffer** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **ShortBuffer** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static **ShortBuffer** * **allocate** (std::size_t capacity)

  Allocates a new Double buffer.

- static **ShortBuffer** * **wrap** (short *array, std::size_t offset, std::size_t length) throw (lang::exceptions::NullPointerException)

  *Wraps the passed buffer with a new **ShortBuffer** (p. 2338).*

- static **ShortBuffer** * **wrap** (std::vector< short > &buffer)

  *Wraps the passed STL short Vector in a **ShortBuffer** (p. 2338).*

Protected Member Functions

- **ShortBuffer** (std::size_t capacity)

  *Creates a **ShortBuffer** (p. 2338) object that has its backing array allocated internally and is then owned and deleted when this object is deleted.*

6.495.1 Detailed Description

This class defines four categories of operations upon short buffers:. o Absolute and relative get and put methods that read and write single shorts; o Relative bulk get methods that transfer contiguous sequences of shorts from this buffer into an array; and o Relative bulk put methods that transfer contiguous sequences of shorts from a short array or some other short buffer into this buffer o Methods for compacting, duplicating, and slicing a short buffer.

Double buffers can be created either by allocation, which allocates space for the buffer's content, by wrapping an existing short array into a buffer, or by creating a view of an existing byte buffer

Methods in this class that do not otherwise have a value to return are specified to return the buffer upon which they are invoked. This allows method invocations to be chained.

6.495.2 Constructor & Destructor Documentation

6.495.2.1 decaf::nio::ShortBuffer::ShortBuffer (std::size_t *capacity*) [protected]

Creates a **ShortBuffer** (p. 2338) object that has its backing array allocated internally and is then owned and deleted when this object is deleted. The array is initially created with all elements initialized to zero.

Parameters:

capacity - size and limit of the **Buffer** (p. 627) in doubles

6.495.2.2 virtual decaf::nio::ShortBuffer::~ShortBuffer () [inline, virtual]

6.495.3 Member Function Documentation

6.495.3.1 static ShortBuffer* decaf::nio::ShortBuffer::allocate (std::size_t *capacity*) [static]

Allocates a new Double buffer. The new buffer's position will be zero, its limit will be its capacity, and its mark will be undefined. It will have a backing array, and its array offset will be zero.

Parameters:

capacity - The size of the Double buffer in shorts

Returns:

the **ShortBuffer** (p. 2338) that was allocated, caller owns.

6.495.3.2 virtual short* decaf::nio::ShortBuffer::array () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the short array that backs this buffer (optional operation). Modifications to this buffer's content will cause the returned array's content to be modified, and vice versa.

Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

the array that backs this **Buffer** (p. 627)

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2332).

6.495.3.3 virtual std::size_t decaf::nio::ShortBuffer::arrayOffset () throw (decaf::lang::exceptions::UnsupportedOperationException, ReadOnlyBufferException) [pure virtual]

Returns the offset within this buffer's backing array of the first element of the buffer (optional operation). Invoke the hasArray method before invoking this method in order to ensure that this buffer has an accessible backing array.

Returns:

The offset into the backing array where index zero starts.

Exceptions:

ReadOnlyBufferException *(p. 2167)* if this **Buffer** (p. 627) is read only.
UnsupportedOperationException if the underlying store has no array.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2333).

6.495.3.4 virtual ShortBuffer* decaf::nio::ShortBuffer::asReadOnlyBuffer () const [pure virtual]

Creates a new, read-only short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer; the new buffer itself, however, will be read-only and will not allow the shared content to be modified. The two buffers' position, limit, and mark values will be independent.

If this buffer is itself read-only then this method behaves in exactly the same way as the duplicate method.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer.

Returns:

The new, read-only short buffer which the caller then owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2333).

6.495.3.5 virtual ShortBuffer& decaf::nio::ShortBuffer::compact () throw (ReadOnlyBufferException) [pure virtual]

Compacts this buffer. The bytes between the buffer's current position and its limit, if any, are copied to the beginning of the buffer. That is, the byte at index p = **position()** (p. 631) is copied to index zero, the byte at index p + 1 is copied to index one, and so forth until the byte at index **limit()** (p. 631) - 1 is copied to index n = **limit()** (p. 631) - 1 - p. The buffer's position is then set to n+1 and its limit is set to its capacity. The mark, if defined, is discarded.

The buffer's position is set to the number of bytes copied, rather than to zero, so that an invocation of this method can be followed immediately by an invocation of another relative put method.

Returns:

a reference to this **ShortBuffer** (p. 2338)

Exceptions:

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2333).

6.495.3.6 virtual int decaf::nio::ShortBuffer::compareTo (const ShortBuffer & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.495.3.7 virtual ShortBuffer* decaf::nio::ShortBuffer::duplicate () [pure virtual]

Creates a new short buffer that shares this buffer's content. The content of the new buffer will be that of this buffer. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's capacity, limit, position, and mark values will be identical to those of this buffer. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

a new short **Buffer** (p. 627) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2334).

6.495.3.8 virtual bool decaf::nio::ShortBuffer::equals (const ShortBuffer & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.495.3.9 ShortBuffer& decaf::nio::ShortBuffer::get (short * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferUnderflowException, lang::exceptions::NullPointerException)

Relative bulk get method. This method transfers shorts from this buffer into the given destination array. If there are fewer shorts remaining in the buffer than are required to satisfy the request, that is, if length > **remaining()** (p. 632), then no bytes are transferred and a **BufferUnderflowException** (p. 661) is thrown.

Otherwise, this method copies length shorts from this buffer into the given array, starting at the current position of this buffer and at the given offset in the array. The position of this buffer is then incremented by length.

Parameters:

buffer - pointer to an allocated buffer to fill

offset - position in the buffer to start filling

length - amount of data to put in the passed buffer

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length shorts remaining in this buffer

NullPointerException if the passed buffer is null.

6.495.3.10 ShortBuffer& decaf::nio::ShortBuffer::get (std::vector< short > *buffer*) throw (BufferUnderflowException)

Relative bulk get method. This method transfers values from this buffer into the given destination vector. An invocation of this method of the form src.get(a) behaves in exactly the same way as the invocation. The vector must be sized to the amount of data that is to be read, that is to say, the caller should call buffer.resize(N) before calling this get method.

Returns:

a reference to this **Buffer** (p. 627)

Exceptions:

BufferUnderflowException *(*p. ***661)*** - If there are fewer than length shorts remaining in this buffer

6.495.3.11 virtual short decaf::nio::ShortBuffer::get (std::size_t *index*) const throw (lang::exceptions::IndexOutOfBoundsException) [pure virtual]

Absolute get method. Reads the value at the given index.

Parameters:

index - the index in the **Buffer** (p. 627) where the short is to be read

Returns:

the short that is located at the given index

Exceptions:

IndexOutOfBoundsException - If index is not smaller than the buffer's limit

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2334).

6.495.3.12 virtual short decaf::nio::ShortBuffer::get () throw (BufferUnderflowException) [pure virtual]

Relative get method. Reads the value at this buffer's current position, and then increments the position.

Returns:

the short at the current position

Exceptions:

BufferUnderflowException *(p. 661)* if there no more data to return

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2335).

6.495.3.13 virtual bool decaf::nio::ShortBuffer::hasArray () const [pure virtual]

Tells whether or not this buffer is backed by an accessible short array. If this method returns true then the array and arrayOffset methods may safely be invoked. Subclasses should override this method if they do not have a backing array as this class always returns true.

Returns:

true if, and only if, this buffer is backed by an array and is not read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2335).

6.495.3.14 virtual bool decaf::nio::ShortBuffer::operator< (const ShortBuffer & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.495.3.15 virtual bool decaf::nio::ShortBuffer::operator== (const ShortBuffer & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.495.3.16 virtual ShortBuffer& decaf::nio::ShortBuffer::put (std::size_t *index*, short *value*) throw (lang::exceptions::IndexOutOfBoundsException, ReadOnlyBufferException) [pure virtual]

Writes the given shorts into this buffer at the given index.

Parameters:

index - position in the **Buffer** (p. 627) to write the data

value - the shorts to write.

Returns:

a reference to this buffer

Exceptions:

IndexOutOfBoundsException - If index greater than the buffer's limit minus the size of the type being written.

ReadOnlyBufferException *(p. 2167)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2335).

6.495.3.17 virtual ShortBuffer& decaf::nio::ShortBuffer::put (short *value*) throw (BufferOverflowException, ReadOnlyBufferException) [pure virtual]

Writes the given shorts into this buffer at the current position, and then increments the position.

Parameters:

value - the shorts value to be written

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. **658**)* - If this buffer's current position is not smaller than its limit

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2336).

6.495.3.18 ShortBuffer& decaf::nio::ShortBuffer::put (std::vector< short > & *buffer*) throw (BufferOverflowException, ReadOnlyBufferException)

This method transfers the entire content of the given source shorts array into this buffer. This is the same as calling put(&buffer[0], 0, buffer.size()

Parameters:

buffer - The buffer whose contents are copied to this **ShortBuffer** (p. 2338)

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. **658**)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

6.495.3.19 ShortBuffer& decaf::nio::ShortBuffer::put (const short * *buffer*, std::size_t *offset*, std::size_t *length*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::NullPointerException)

This method transfers shorts into this buffer from the given source array. If there are more shorts to be copied from the array than remain in this buffer, that is, if length > **remaining()** (p. 632), then no shorts are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies length bytes from the given array into this buffer, starting at the given offset in the array and at the current position of this buffer. The position of this buffer is then incremented by length.

Parameters:

buffer- The array from which shorts are to be read

offset- The offset within the array of the first short to be read;

length - The number of shorts to be read from the given array

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(p. **658**)* - If there is insufficient space in this buffer

ReadOnlyBufferException *(p. **2167**)* - If this buffer is read-only

NullPointerException if the passed buffer is null.

6.495.3.20 ShortBuffer& decaf::nio::ShortBuffer::put (ShortBuffer & *src*) throw (BufferOverflowException, ReadOnlyBufferException, lang::exceptions::IllegalArgumentException)

This method transfers the shorts remaining in the given source buffer into this buffer. If there are more shorts remaining in the source buffer than in this buffer, that is, if src.remaining() > **remaining()** (p. 632), then no shorts are transferred and a **BufferOverflowException** (p. 658) is thrown.

Otherwise, this method copies n = src.remaining() shorts from the given buffer into this buffer, starting at each buffer's current position. The positions of both buffers are then incremented by n.

Parameters:

src - the buffer to take shorts from an place in this one.

Returns:

a reference to this buffer

Exceptions:

BufferOverflowException *(*p. ***658****)* - If there is insufficient space in this buffer for the remaining shorts in the source buffer

IllegalArgumentException - If the source buffer is this buffer

ReadOnlyBufferException *(*p. ***2167****)* - If this buffer is read-only

6.495.3.21 virtual ShortBuffer* decaf::nio::ShortBuffer::slice () const [pure virtual]

Creates a new **ShortBuffer** (p. 2338) whose content is a shared subsequence of this buffer's content. The content of the new buffer will start at this buffer's current position. Changes to this buffer's content will be visible in the new buffer, and vice versa; the two buffers' position, limit, and mark values will be independent.

The new buffer's position will be zero, its capacity and its limit will be the number of bytes remaining in this buffer, and its mark will be undefined. The new buffer will be read-only if, and only if, this buffer is read-only.

Returns:

the newly create **ShortBuffer** (p. 2338) which the caller owns.

Implemented in **decaf::internal::nio::ShortArrayBuffer** (p. 2336).

6.495.3.22 virtual std::string decaf::nio::ShortBuffer::toString () const [virtual]

Returns:

a std::string describing this object

6.495.3.23 static ShortBuffer* decaf::nio::ShortBuffer::wrap (std::vector< short > & *buffer*) [static]

Wraps the passed STL short Vector in a **ShortBuffer** (p. 2338). The new buffer will be backed by the given short array; modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity and limit will be buffer.size(), its position will be zero, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

buffer - The vector that will back the new buffer, the vector must have been sized to the desired size already by calling vector.resize(N).

Returns:

a new **ShortBuffer** (p. 2338) that is backed by buffer, caller owns.

6.495.3.24 static ShortBuffer* decaf::nio::ShortBuffer::wrap (short * *array*, std::size_t *offset*, std::size_t *length*) throw (lang::exceptions::NullPointerException) [static]

Wraps the passed buffer with a new **ShortBuffer** (p. 2338). The new buffer will be backed by the given short array; that is, modifications to the buffer will cause the array to be modified and vice versa. The new buffer's capacity will be array.length, its position will be offset, its limit will be offset + length, and its mark will be undefined. Its backing array will be the given array, and its array offset will be zero.

Parameters:

array - The array that will back the new buffer

offset - The offset of the subarray to be used

length - The length of the subarray to be used

Returns:

a new **ShortBuffer** (p. 2338) that is backed by buffer, caller owns.

The documentation for this class was generated from the following file:

- src/main/decaf/nio/**ShortBuffer.h**

6.496 activemq::commands::ShutdownInfo Class Reference

`#include <src/main/activemq/commands/ShutdownInfo.h>`Inheritance diagram for activemq::commands::ShutdownInfo:

Public Member Functions

- **ShutdownInfo** ()
- virtual ~**ShutdownInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **ShutdownInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual bool **isShutdownInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_SHUTDOWNINFO** = 11

Protected Member Functions

- **ShutdownInfo** (const **ShutdownInfo** &)
- **ShutdownInfo** & **operator**= (const **ShutdownInfo** &)

6.496.1 Constructor & Destructor Documentation

6.496.1.1 activemq::commands::ShutdownInfo::ShutdownInfo (const ShutdownInfo &) [inline, protected]

6.496.1.2 activemq::commands::ShutdownInfo::ShutdownInfo ()

6.496.1.3 virtual activemq::commands::ShutdownInfo::~ShutdownInfo () [virtual]

6.496.2 Member Function Documentation

6.496.2.1 virtual ShutdownInfo* activemq::commands::ShutdownInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.496.2.2 virtual void activemq::commands::ShutdownInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.496.2.3 virtual bool activemq::commands::ShutdownInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.496.2.4 virtual unsigned char activemq::commands::ShutdownInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.496.2.5 virtual bool activemq::commands::ShutdownInfo::isShutdownInfo () const [inline, virtual]

Returns:

an answer of true to the **isShutdownInfo()** (p. 2351) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 511).

6.496.2.6 ShutdownInfo& activemq::commands::ShutdownInfo::operator= (const ShutdownInfo &) [inline, protected]

6.496.2.7 virtual std::string activemq::commands::ShutdownInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.496.2.8 virtual Pointer<Command> activemq::commands::ShutdownInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.496.3 Field Documentation

6.496.3.1 const unsigned char activemq::commands::ShutdownInfo::ID_SHUTDOWNINFO = 11 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**ShutdownInfo.h**

6.497 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2352).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/ShutdownInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual ~**ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.497.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2352). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.497.2 Constructor & Destructor Documentation

6.497.2.1 activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::ShutdownInfoMars () [inline]

6.497.2.2 virtual activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::~ShutdownInfoMa () [inline, virtual]

6.497.3 Member Function Documentation

6.497.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.497.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::getDataStructureTyp () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.497.3.3 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.497.3.4 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.497.3.5 virtual int activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.497.3.6 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.497.3.7 virtual void activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**ShutdownInfoMarshaller.h**

6.498 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2356).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/ShutdownInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual ~**ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.498.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2356). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.498.2 Constructor & Destructor Documentation

6.498.2.1 activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::ShutdownInfoMars () [inline]

6.498.2.2 virtual activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::~ShutdownInfoMa () [inline, virtual]

6.498.3 Member Function Documentation

6.498.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.498.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.498.3.3 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.498.3.4 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.498.3.5 virtual int activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.498.3.6 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.498.3.7 virtual void activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**ShutdownInfoMarshaller.h**

6.499 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarsh Class Reference

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2360).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/ShutdownInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller:

Public Member Functions

- **ShutdownInfoMarshaller** ()
- virtual ~**ShutdownInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.499.1 Detailed Description

Marshaling code for Open Wire Format for **ShutdownInfoMarshaller** (p. 2360). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.499.2 Constructor & Destructor Documentation

6.499.2.1 activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::ShutdownInfoMars () [inline]

6.499.2.2 virtual activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::~ShutdownInfoMa () [inline, virtual]

6.499.3 Member Function Documentation

6.499.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.499.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::getDataStructureTy () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.499.3.3 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.499.3.4 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.499.3.5 virtual int activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.499.3.6 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.499.3.7 virtual void activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**ShutdownInfoMarshaller.h**

6.500 decaf::security::SignatureException Class Reference

`#include <src/main/decaf/security/SignatureException.h>`Inheritance diagram for decaf::security::SignatureException:

Public Member Functions

- **SignatureException** () throw ()
  Default Constructor.
- **SignatureException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **SignatureException** (const **SignatureException** &ex) throw ()
  Copy Constructor.
- **SignatureException** (const char *file, const int lineNumber, const std::exception *cause, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **SignatureException** (const std::exception *cause) throw ()
  Constructor.
- **SignatureException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **SignatureException** * **clone** () const
  Clones this exception.
- virtual ~**SignatureException** () throw ()

6.500.1 Constructor & Destructor Documentation

6.500.1.1 decaf::security::SignatureException::SignatureException () throw () [inline]

Default Constructor.

6.500.1.2 decaf::security::SignatureException::SignatureException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.500.1.3 decaf::security::SignatureException::SignatureException (const SignatureException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.500.1.4 decaf::security::SignatureException::SignatureException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.500.1.5 decaf::security::SignatureException::SignatureException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.500.1.6 decaf::security::SignatureException::SignatureException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file name where exception occurs

lineNumber line number where the exception occurred.

msg message to report

... list of primitives that are formatted into the message

6.500.1.7 virtual decaf::security::SignatureException::~SignatureException () throw () [inline, virtual]

6.500.2 Member Function Documentation

6.500.2.1 virtual SignatureException* decaf::security::SignatureException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A deep copy of this exception.

Reimplemented from **decaf::security::GeneralSecurityException** (p. 1381).

The documentation for this class was generated from the following file:

- src/main/decaf/security/**SignatureException.h**

6.501 decaf::util::logging::SimpleFormatter Class Reference

Print a brief summary of the **LogRecord** (p. 1645) in a human readable format.

`#include <src/main/decaf/util/logging/SimpleFormatter.h>`Inheritance diagram for decaf::util::logging::SimpleFormatter:

Public Member Functions

- **SimpleFormatter** ()
- virtual ~**SimpleFormatter** ()
- virtual std::string **format** (const **LogRecord** &record) const

  Format the given log record and return the formatted string.
- virtual std::string **formatMessage** (const **LogRecord** &record) const

  Format the message string from a log record.
- virtual std::string **getHead** (const **Handler** *handler)

  Return the header string for a set of formatted records.
- virtual std::string **getTail** (const **Handler** *handler)

  Return the tail string for a set of formatted records.

6.501.1 Detailed Description

Print a brief summary of the **LogRecord** (p. 1645) in a human readable format. The summary will typically be 1 or 2 lines.

6.501.2 Constructor & Destructor Documentation

6.501.2.1 decaf::util::logging::SimpleFormatter::SimpleFormatter () `[inline]`

6.501.2.2 virtual decaf::util::logging::SimpleFormatter::~SimpleFormatter () `[inline, virtual]`

6.501.3 Member Function Documentation

6.501.3.1 virtual std::string decaf::util::logging::SimpleFormatter::format (const LogRecord & *record*) const `[inline, virtual]`

Format the given log record and return the formatted string.

Parameters:

record The Log Record to Format

Implements **decaf::util::logging::Formatter** (p. 1372).

6.501.3.2 virtual std::string decaf::util::logging::SimpleFormatter::formatMessage (const LogRecord & *record*) const [inline, virtual]

Format the message string from a log record.

Parameters:

record The Log Record to Format

Implements **decaf::util::logging::Formatter** (p. 1373).

References decaf::util::logging::LogRecord::getMessage().

6.501.3.3 virtual std::string decaf::util::logging::SimpleFormatter::getHead (const Handler * *handler*) [inline, virtual]

Return the header string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

empty string

Implements **decaf::util::logging::Formatter** (p. 1373).

6.501.3.4 virtual std::string decaf::util::logging::SimpleFormatter::getTail (const Handler * *handler*) [inline, virtual]

Return the tail string for a set of formatted records. In the default implementation this method should return empty string

Parameters:

handler the target handler, can be null

Returns:

empty string

Implements **decaf::util::logging::Formatter** (p. 1373).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleFormatter.h**

6.502 decaf::util::logging::SimpleLogger Class Reference

`#include <src/main/decaf/util/logging/SimpleLogger.h>`

Public Member Functions

- **SimpleLogger** (const std::string &name)

  Constructor.

- virtual ~**SimpleLogger** ()

  Destructor.

- virtual void **mark** (const std::string &message)

  Log a Mark Block Level Log.

- virtual void **debug** (const std::string &file, const int line, const std::string &message)

  Log a Debug Level Log.

- virtual void **info** (const std::string &file, const int line, const std::string &message)

  Log a Informational Level Log.

- virtual void **warn** (const std::string &file, const int line, const std::string &message)

  Log a Warning Level Log.

- virtual void **error** (const std::string &file, const int line, const std::string &message)

  Log a Error Level Log.

- virtual void **fatal** (const std::string &file, const int line, const std::string &message)

  Log a Fatal Level Log.

- virtual void **log** (const std::string &message)

  No-frills log.

6.502.1 Constructor & Destructor Documentation

6.502.1.1 decaf::util::logging::SimpleLogger::SimpleLogger (const std::string & *name*)

Constructor.

6.502.1.2 virtual decaf::util::logging::SimpleLogger::~SimpleLogger () [virtual]

Destructor.

6.502.2 Member Function Documentation

6.502.2.1 virtual void decaf::util::logging::SimpleLogger::debug (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Debug Level Log.

6.502.2.2 virtual void decaf::util::logging::SimpleLogger::error (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Error Level Log.

6.502.2.3 virtual void decaf::util::logging::SimpleLogger::fatal (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Fatal Level Log.

6.502.2.4 virtual void decaf::util::logging::SimpleLogger::info (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Informational Level Log.

6.502.2.5 virtual void decaf::util::logging::SimpleLogger::log (const std::string & *message*) [virtual]

No-frills log.

6.502.2.6 virtual void decaf::util::logging::SimpleLogger::mark (const std::string & *message*) [virtual]

Log a Mark Block Level Log.

6.502.2.7 virtual void decaf::util::logging::SimpleLogger::warn (const std::string & *file*, const int *line*, const std::string & *message*) [virtual]

Log a Warning Level Log.

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**SimpleLogger.h**

6.503 decaf::net::Socket Class Reference

`#include <src/main/decaf/net/Socket.h>`Inheritance diagram for decaf::net::Socket:

Public Types

- typedef apr_socket_t * **SocketHandle**
  Define the SocketHandle type.
- typedef apr_sockaddr_t * **SocketAddress**
  Define the SocketAddress type.

Public Member Functions

- virtual ~**Socket** ()
- virtual void **connect** (const char *host, int port)=0 throw (SocketException)
  Connects to the specified destination.
- virtual bool **isConnected** () const =0
  Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * **getInputStream** ()=0
  Gets the InputStream for this socket.
- virtual **io::OutputStream** * **getOutputStream** ()=0
  Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const =0 throw (SocketException)
  Gets the linger time.
- virtual void **setSoLinger** (int linger)=0 throw (SocketException)
  Sets the linger time.
- virtual bool **getKeepAlive** () const =0 throw (SocketException)
  Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive)=0 throw (SocketException)
  Enables/disables the keep alive flag.
- virtual int **getReceiveBufferSize** () const =0 throw (SocketException)
  Gets the receive buffer size.
- virtual void **setReceiveBufferSize** (int size)=0 throw (SocketException)
  Sets the receive buffer size.
- virtual bool **getReuseAddress** () const =0 throw (SocketException)

Gets the reuse address flag.

- virtual void **setReuseAddress** (bool reuse)=0 throw (SocketException)
  Sets the reuse address flag.
- virtual int **getSendBufferSize** () const =0 throw (SocketException)
  Gets the send buffer size.
- virtual void **setSendBufferSize** (int size)=0 throw (SocketException)
  Sets the send buffer size.
- virtual int **getSoTimeout** () const =0 throw (SocketException)
  Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout)=0 throw (SocketException)
  Sets the timeout for socket operations.

Static Public Attributes

- static const int **INVALID_SOCKET_HANDLE** = 0
  Defines a constant for an invalid socket handle.

6.503.1 Member Typedef Documentation

6.503.1.1 typedef apr_sockaddr_t* decaf::net::Socket::SocketAddress

Define the SocketAddress type.

6.503.1.2 typedef apr_socket_t* decaf::net::Socket::SocketHandle

Define the SocketHandle type.

6.503.2 Constructor & Destructor Documentation

6.503.2.1 virtual decaf::net::Socket::~Socket () [inline, virtual]

6.503.3 Member Function Documentation

6.503.3.1 virtual void decaf::net::Socket::connect (const char * *host*, int *port*) throw (SocketException) [pure virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

IOException Thrown if a failure occurred in the connect.

Implemented in **decaf::net::BufferedSocket** (p. 643), and **decaf::net::TcpSocket** (p. 2526).

6.503.3.2 virtual io::InputStream* decaf::net::Socket::getInputStream () [pure virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implemented in **decaf::net::BufferedSocket** (p. 643), and **decaf::net::TcpSocket** (p. 2527).

6.503.3.3 virtual bool decaf::net::Socket::getKeepAlive () const throw (SocketException) [pure virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 643), and **decaf::net::TcpSocket** (p. 2527).

Referenced by decaf::net::BufferedSocket::getKeepAlive().

6.503.3.4 virtual io::OutputStream* decaf::net::Socket::getOutputStream () [pure virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implemented in **decaf::net::BufferedSocket** (p. 643), and **decaf::net::TcpSocket** (p. 2527).

6.503.3.5 virtual int decaf::net::Socket::getReceiveBufferSize () const throw (SocketException) [pure virtual]

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 644), and **decaf::net::TcpSocket** (p. 2527).

Referenced by decaf::net::BufferedSocket::getReceiveBufferSize().

6.503.3.6 virtual bool decaf::net::Socket::getReuseAddress () const throw (SocketException) [pure virtual]

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 644), and **decaf::net::TcpSocket** (p. 2528).

Referenced by decaf::net::BufferedSocket::getReuseAddress().

6.503.3.7 virtual int decaf::net::Socket::getSendBufferSize () const throw (SocketException) [pure virtual]

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 644), and **decaf::net::TcpSocket** (p. 2528).

Referenced by decaf::net::BufferedSocket::getSendBufferSize().

6.503.3.8 virtual int decaf::net::Socket::getSoLinger () const throw (SocketException) [pure virtual]

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 644), and **decaf::net::TcpSocket** (p. 2528).

Referenced by decaf::net::BufferedSocket::getSoLinger().

6.503.3.9 virtual int decaf::net::Socket::getSoTimeout () const throw (SocketException) [pure virtual]

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(p. 2379)* Thrown if unable to retrieve the information.

Implemented in **decaf::net::BufferedSocket** (p. 645), and **decaf::net::TcpSocket** (p. 2529).

Referenced by decaf::net::BufferedSocket::getSoTimeout().

6.503.3.10 virtual bool decaf::net::Socket::isConnected () const [pure virtual]

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implemented in **decaf::net::BufferedSocket** (p. 645), and **decaf::net::TcpSocket** (p. 2529).

Referenced by decaf::net::BufferedSocket::isConnected().

6.503.3.11 virtual void decaf::net::Socket::setKeepAlive (bool *keepAlive*) throw (SocketException) [pure virtual]

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException *(p. 2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 645), and **decaf::net::TcpSocket** (p. 2529).

Referenced by decaf::net::BufferedSocket::setKeepAlive().

6.503.3.12 virtual void decaf::net::Socket::setReceiveBufferSize (int *size*) throw (SocketException) [pure virtual]

Sets the receive buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 646), and **decaf::net::TcpSocket** (p. 2530).

Referenced by decaf::net::BufferedSocket::setReceiveBufferSize().

6.503.3.13 virtual void decaf::net::Socket::setReuseAddress (bool *reuse*) throw (SocketException) [pure virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 646), and **decaf::net::TcpSocket** (p. 2530).

Referenced by decaf::net::BufferedSocket::setReuseAddress().

6.503.3.14 virtual void decaf::net::Socket::setSendBufferSize (int *size*) throw (SocketException) [pure virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 646), and **decaf::net::TcpSocket** (p. 2530).

Referenced by decaf::net::BufferedSocket::setSendBufferSize().

6.503.3.15 virtual void decaf::net::Socket::setSoLinger (int *linger*) throw (SocketException) [pure virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implemented in **decaf::net::BufferedSocket** (p. 646), and **decaf::net::TcpSocket** (p. 2530).

Referenced by decaf::net::BufferedSocket::setSoLinger().

6.503.3.16 virtual void decaf::net::Socket::setSoTimeout (int *timeout*) throw (SocketException) [pure virtual]

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(*p. *2379)* Thrown if unable to set the information.

Implemented in **decaf::net::BufferedSocket** (p. 647), and **decaf::net::TcpSocket** (p. 2531).

Referenced by decaf::net::BufferedSocket::setSoTimeout().

6.503.4 Field Documentation

6.503.4.1 const int decaf::net::Socket::INVALID_SOCKET_HANDLE = 0 [static]

Defines a constant for an invalid socket handle.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**Socket.h**

6.504 decaf::net::SocketError Class Reference

Static utility class to simplify handling of error codes for socket operations.

`#include <src/main/decaf/net/SocketError.h>`

Static Public Member Functions

- static int **getErrorCode** ()
  Gets the last error appropriate for the platform.
- static std::string **getErrorString** ()
  Gets the string description for the last error.

6.504.1 Detailed Description

Static utility class to simplify handling of error codes for socket operations.

6.504.2 Member Function Documentation

6.504.2.1 static int decaf::net::SocketError::getErrorCode () [static]

Gets the last error appropriate for the platform.

6.504.2.2 static std::string decaf::net::SocketError::getErrorString () [static]

Gets the string description for the last error.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketError.h**

6.505 decaf::net::SocketException Class Reference

Exception for errors when manipulating sockets.

`#include <src/main/decaf/net/SocketException.h>`Inheritance diagram for decaf::net::SocketException:

Public Member Functions

- **SocketException** () throw ()
- **SocketException** (const **lang::Exception** &ex) throw ()
- **SocketException** (const **SocketException** &ex) throw ()
- **SocketException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- **SocketException** (const std::exception ***cause**) throw ()

  Constructor.

- **SocketException** (const char *file, const int lineNumber, const char *msg,...) throw ()

  Constructor - Initializes the file name and line number where this message occurred.

- virtual **SocketException** * **clone** () const

  Clones this exception.

- virtual ~**SocketException** () throw ()

6.505.1 Detailed Description

Exception for errors when manipulating sockets.

6.505.2 Constructor & Destructor Documentation

6.505.2.1 decaf::net::SocketException::SocketException () throw () [inline]

6.505.2.2 decaf::net::SocketException::SocketException (const lang::Exception & *ex*) throw () [inline]

6.505.2.3 decaf::net::SocketException::SocketException (const SocketException & *ex*) throw () [inline]

6.505.2.4 decaf::net::SocketException::SocketException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.505.2.5 decaf::net::SocketException::SocketException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.505.2.6 decaf::net::SocketException::SocketException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.505.2.7 virtual decaf::net::SocketException::~SocketException () throw () [inline, virtual]

6.505.3 Member Function Documentation

6.505.3.1 virtual SocketException* decaf::net::SocketException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

Reimplemented in **decaf::net::BindException** (p. 565), **decaf::net::ConnectException** (p. 940), **decaf::net::NoRouteToHostException** (p. 1941), and **decaf::net::PortUnreachableException** (p. 2039).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketException.h**

6.506 decaf::net::SocketFactory Class Reference

Socket (p. 2371) Factory implementation for use in Creating Sockets.

`#include <src/main/decaf/net/SocketFactory.h>`

Public Member Functions

- virtual ~**SocketFactory** ()

Static Public Member Functions

- static **Socket** * **createSocket** (const std::string &uri, const **util::Properties** &properties) throw (SocketException)

  *Creates and returns a **Socket** (p. 2371) derived Object based on the values defined in the Properties Object that is passed in.*

6.506.1 Detailed Description

Socket (p. 2371) Factory implementation for use in Creating Sockets. Property Options:

Name Value

---

inputBufferSize size in bytes of the buffered input stream buffer. Defaults to 10000.

outputBufferSize size in bytes of the buffered output stream buffer. Defaults to 10000.

soLinger linger time for the socket (in microseconds). Defaults to 0.

soKeepAlive keep alive flag for the socket (true/false). Defaults to false.

soReceiveBufferSize The size of the socket receive buffer (in bytes). Defaults to 2MB.

soSendBufferSize The size of the socket send buffer (in bytes). Defaults to 2MB.

soTimeout The timeout of socket IO operations (in microseconds). Defaults to 10000

See also:

`Socket (p. 2371)`

6.506.2 Constructor & Destructor Documentation

6.506.2.1 virtual decaf::net::SocketFactory::~SocketFactory () [inline, virtual]

6.506.3 Member Function Documentation

6.506.3.1 static Socket* decaf::net::SocketFactory::createSocket (const std::string & *uri*, const util::Properties & *properties*) throw (SocketException) [static]

Creates and returns a **Socket** (p. 2371) derived Object based on the values defined in the Properties Object that is passed in.

Parameters:

uri the **URI** (p. 2638) for the **Socket** (p. 2371) Connection.

properties A Properties object that contains configuration details.

Exceptions:

SocketException.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketFactory.h**

6.507 decaf::net::SocketInputStream Class Reference

Input stream for performing reads on a socket.

`#include <src/main/decaf/net/SocketInputStream.h>`Inheritance diagram for decaf::net::SocketInputStream:

Public Member Functions

- **SocketInputStream** (**Socket::SocketHandle** socket)

  Constructor.

- virtual ~**SocketInputStream** ()

  Destructor.

- virtual std::size_t **available** () const throw (io::IOException)

  Returns the number of bytes available on the socket to be read right now.

- virtual unsigned char **read** () throw (io::IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (io::IOException, lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual void **close** () throw (lang::Exception)

  Close - does nothing.

- virtual std::size_t **skip** (std::size_t num) throw (io::IOException, lang::exceptions::UnsupportedOperationException)

  Not supported.

- virtual void **mark** (int readLimit DECAF_UNUSED)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (io::IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

  Determines if this input stream supports the mark and reset methods.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.507.1 Detailed Description

Input stream for performing reads on a socket. This class will only work properly for blocking sockets.

6.507.2 Constructor & Destructor Documentation

6.507.2.1 decaf::net::SocketInputStream::SocketInputStream (Socket::SocketHandle *socket*)

Constructor.

Parameters:

socket the socket handle.

6.507.2.2 virtual decaf::net::SocketInputStream::~SocketInputStream () [virtual]

Destructor.

6.507.3 Member Function Documentation

6.507.3.1 virtual std::size_t decaf::net::SocketInputStream::available () const throw (io::IOException) [virtual]

Returns the number of bytes available on the socket to be read right now.

Returns:

The number of bytes currently available to be read on the socket.

Implements **decaf::io::InputStream** (p. 1407).

6.507.3.2 virtual void decaf::net::SocketInputStream::close () throw (lang::Exception) [virtual]

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Exceptions:

CMSException

Implements **decaf::io::Closeable** (p. 840).

6.507.3.3 virtual void decaf::net::SocketInputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.507.3.4 virtual void decaf::net::SocketInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1407).

6.507.3.5 virtual bool decaf::net::SocketInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1407).

6.507.3.6 virtual void decaf::net::SocketInputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.507.3.7 virtual void decaf::net::SocketInputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.507.3.8 virtual int decaf::net::SocketInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer. If the desired amount of data is not currently available, this operation will block until the appropriate amount of data is available.

Parameters:

buffer (out) the target buffer

offset the position in the buffer to start from.

bufferSize the size of the output buffer.

Returns:

the number of bytes read. or -1 if EOF

Exceptions:

IOException f an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.507.3.9 virtual unsigned char decaf::net::SocketInputStream::read () throw (io::IOException) [virtual]

Reads a single byte from the buffer. If no data is available, blocks until their is.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.507.3.10 virtual void decaf::net::SocketInputStream::reset () throw (io::IOException) [inline, virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: ∗ If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an IOException might be thrown. ∗ If such an IOException is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: ∗ The call to reset may throw an IOException. ∗ If an IOException is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException

Implements **decaf::io::InputStream** (p. 1408).

6.507.3.11 virtual std::size_t decaf::net::SocketInputStream::skip (std::size_t *num*) throw (io::IOException, lang::exceptions::UnsupportedOperationException) [virtual]

Not supported.

Exceptions:

an UnsupportedOperationException.

Implements **decaf::io::InputStream** (p. 1409).

6.507.3.12 virtual void decaf::net::SocketInputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.507.3.13 virtual void decaf::net::SocketInputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.507.3.14 virtual void decaf::net::SocketInputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketInputStream.h**

6.508 decaf::net::SocketOutputStream Class Reference

Output stream for performing write operations on a socket.

`#include <src/main/decaf/net/SocketOutputStream.h>`Inheritance diagram for decaf::net::SocketOutputStream:

Public Member Functions

- **SocketOutputStream** (**Socket::SocketHandle** socket)

  Constructor.

- virtual ~**SocketOutputStream** ()
- virtual void **write** (unsigned char c) throw (io::IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (io::IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (io::IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (io::IOException)

  Flush - does nothing.

- virtual void **close** () throw (lang::Exception)

  Close - does nothing.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.508.1 Detailed Description

Output stream for performing write operations on a socket.

6.508.2 Constructor & Destructor Documentation

6.508.2.1 decaf::net::SocketOutputStream::SocketOutputStream (Socket::SocketHandle *socket*)

Constructor.

Parameters:

socket the socket handle.

6.508.2.2 virtual decaf::net::SocketOutputStream::~SocketOutputStream () [virtual]

6.508.3 Member Function Documentation

6.508.3.1 virtual void decaf::net::SocketOutputStream::close () throw (lang::Exception) [virtual]

Close - does nothing. It is the responsibility of the owner of the socket object to close it.

Exceptions:

CMSException

Implements **decaf::io::Closeable** (p. 840).

6.508.3.2 virtual void decaf::net::SocketOutputStream::flush () throw (io::IOException) [inline, virtual]

Flush - does nothing.

Exceptions:

IOException

Implements **decaf::io::OutputStream** (p. 1992).

6.508.3.3 virtual void decaf::net::SocketOutputStream::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.508.3.4 virtual void decaf::net::SocketOutputStream::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.508.3.5 virtual void decaf::net::SocketOutputStream::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.508.3.6 virtual void decaf::net::SocketOutputStream::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.508.3.7 virtual void decaf::net::SocketOutputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.508.3.8 virtual void decaf::net::SocketOutputStream::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.508.3.9 virtual void decaf::net::SocketOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (io::IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

IOException thrown if an error occurs.

NullPointerException thrown if buffer is Null.

Implements **decaf::io::OutputStream** (p. 1993).

6.508.3.10 virtual void decaf::net::SocketOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (io::IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

6.508.3.11 virtual void decaf::net::SocketOutputStream::write (unsigned char *c*) throw (io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketOutputStream.h**

6.509 decaf::net::SocketTimeoutException Class Reference

`#include <src/main/decaf/net/SocketTimeoutException.h>`Inheritance diagram for decaf::net::SocketTimeoutException:

Public Member Functions

- **SocketTimeoutException** () throw ()
  Default Constructor.
- **SocketTimeoutException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **SocketTimeoutException** (const **SocketTimeoutException** &ex) throw ()
  Copy Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **SocketTimeoutException** (const std::exception ***cause**) throw ()
  Constructor.
- **SocketTimeoutException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **SocketTimeoutException** * **clone** () const
  Clones this exception.
- virtual ~**SocketTimeoutException** () throw ()

6.509.1 Constructor & Destructor Documentation

6.509.1.1 decaf::net::SocketTimeoutException::SocketTimeoutException () throw () `[inline]`

Default Constructor.

6.509.1.2 decaf::net::SocketTimeoutException::SocketTimeoutException (const Exception & *ex*) throw () `[inline]`

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.509.1.3 decaf::net::SocketTimeoutException::SocketTimeoutException (const SocketTimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.509.1.4 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.509.1.5 decaf::net::SocketTimeoutException::SocketTimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.509.1.6 decaf::net::SocketTimeoutException::SocketTimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.509.1.7 virtual decaf::net::SocketTimeoutException::~SocketTimeoutException () throw () [inline, virtual]

6.509.2 Member Function Documentation

6.509.2.1 virtual SocketTimeoutException* decaf::net::SocketTimeoutException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::InterruptedIOException** (p. 1464).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**SocketTimeoutException.h**

6.510 activemq::commands::BrokerError::StackTraceElement Struct Reference

`#include <src/main/activemq/commands/BrokerError.h>`

Data Fields

- std::string **ClassName**
- std::string **FileName**
- std::string **MethodName**
- int **LineNumber**

6.510.1 Field Documentation

6.510.1.1 std::string activemq::commands::BrokerError::StackTraceElement::ClassName

6.510.1.2 std::string activemq::commands::BrokerError::StackTraceElement::FileName

6.510.1.3 int activemq::commands::BrokerError::StackTraceElement::LineNumber

6.510.1.4 std::string activemq::commands::BrokerError::StackTraceElement::MethodName

The documentation for this struct was generated from the following file:

- src/main/activemq/commands/**BrokerError.h**

6.511 decaf::internal::io::StandardErrorOutputStream Class Reference

Wrapper Around the Standard error Output facility on the current platform.

`#include <src/main/decaf/internal/io/StandardErrorOutputStream.h>`Inheritance diagram for decaf::internal::io::StandardErrorOutputStream:

Public Member Functions

- **StandardErrorOutputStream** ()

  Default Constructor.

- virtual ~**StandardErrorOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (decaf::io::IOException)

  Invokes flush on the target output stream.

- virtual void **close** () throw (decaf::lang::Exception)

  Invokes close on the target output stream.

- virtual void **lock** () throw (decaf::lang::Exception)

  Locks the object.

- virtual void **wait** () throw (decaf::lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (decaf::lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.511.1 Detailed Description

Wrapper Around the Standard error Output facility on the current platform. This allows for the use of alternate output methods on platforms or compilers that do not support `std::cerr`.

6.511.2 Constructor & Destructor Documentation

6.511.2.1 decaf::internal::io::StandardErrorOutputStream::StandardErrorOutputStream ()

Default Constructor.

6.511.2.2 virtual decaf::internal::io::StandardErrorOutputStream::~StandardErrorOutputStream () [virtual]

6.511.3 Member Function Documentation

6.511.3.1 virtual void decaf::internal::io::StandardErrorOutputStream::close () throw (decaf::lang::Exception) [inline, virtual]

Invokes close on the target output stream. throws CMSException if an error occurs

Implements **decaf::io::Closeable** (p. 840).

6.511.3.2 virtual void decaf::internal::io::StandardErrorOutputStream::flush () throw (decaf::io::IOException) [virtual]

Invokes flush on the target output stream. throws **decaf::io::IOException** (p. 1477) if an error occurs

Implements **decaf::io::OutputStream** (p. 1992).

6.511.3.3 virtual void decaf::internal::io::StandardErrorOutputStream::lock () throw (decaf::lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.511.3.4 virtual void decaf::internal::io::StandardErrorOutputStream::notify () throw (decaf::lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.511.3.5 virtual void decaf::internal::io::StandardErrorOutputStream::notifyAll () throw (decaf::lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.511.3.6 virtual void decaf::internal::io::StandardErrorOutputStream::wait (unsigned long *millisecs*) throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.511.3.7 virtual void decaf::internal::io::StandardErrorOutputStream::wait () throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.511.3.8 virtual void decaf::internal::io::StandardErrorOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (decaf::io::IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset The position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

decaf::io::IOException *(*p. ***1477)*** thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::OutputStream** (p. 1993).

6.511.3.9 virtual void decaf::internal::io::StandardErrorOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (decaf::io::IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

6.511.3.10 virtual void decaf::internal::io::StandardErrorOutputStream::write (unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardErrorOutputStream.h**

6.512 decaf::internal::io::StandardInputStream Class Reference

`#include <src/main/decaf/internal/io/StandardInputStream.h>`Inheritance diagram for decaf::internal::io::StandardInputStream:

Public Member Functions

- **StandardInputStream** ()
- virtual ~**StandardInputStream** ()
- virtual std::size_t **available** () const throw (decaf::io::IOException)

  Indicates the number of bytes available.

- virtual unsigned char **read** () throw (decaf::io::IOException)

  Reads a single byte from the buffer.

- virtual int **read** (unsigned char *buffer, std::size_t offset, std::size_t bufferSize) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException)

  Reads an array of bytes from the buffer.

- virtual void **close** () throw (lang::Exception)

  Closes the target input stream.

- virtual std::size_t **skip** (std::size_t num) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Skips over and discards n bytes of data from this input stream.

- virtual void **mark** (int readLimit DECAF_UNUSED)

  Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes.

- virtual void **reset** () throw (decaf::io::IOException)

  Repositions this stream to the position at the time the mark method was last called on this input stream.

- virtual bool **markSupported** () const

  Determines if this input stream supports the mark and reset methods.

Protected Member Functions

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.512.1 Constructor & Destructor Documentation

6.512.1.1 decaf::internal::io::StandardInputStream::StandardInputStream ()

6.512.1.2 virtual decaf::internal::io::StandardInputStream::~StandardInputStream () [virtual]

6.512.2 Member Function Documentation

6.512.2.1 virtual std::size_t decaf::internal::io::StandardInputStream::available () const throw (decaf::io::IOException) [virtual]

Indicates the number of bytes available.

Returns:

The number of bytes until the end of the **internal** (p. 95) buffer.

Implements **decaf::io::InputStream** (p. 1407).

6.512.2.2 virtual void decaf::internal::io::StandardInputStream::close () throw (lang::Exception) [inline, virtual]

Closes the target input stream.

Exceptions:

decaf::io::IOException (p. ***1477***) thrown if an error occurs.

Implements **decaf::io::Closeable** (p. 840).

6.512.2.3 virtual void decaf::internal::io::StandardInputStream::lock () throw (lang::Exception) [inline, protected, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.512.2.4 virtual void decaf::internal::io::StandardInputStream::mark (int readLimit *DECAF_UNUSED*) [inline, virtual]

Marks the current position in the stream A subsequent call to the reset method repositions this stream at the last marked position so that subsequent reads re-read the same bytes. If a stream instance reports that marks are supported then the stream will ensure that the same bytes can be read again after the reset method is called so long the readLimit is not reached.

Parameters:

readLimit - max bytes read before marked position is invalid.

Implements **decaf::io::InputStream** (p. 1407).

6.512.2.5 virtual bool decaf::internal::io::StandardInputStream::markSupported () const [inline, virtual]

Determines if this input stream supports the mark and reset methods. Whether or not mark and reset are supported is an invariant property of a particular input stream instance.

Returns:

true if this stream instance supports marks

Implements **decaf::io::InputStream** (p. 1407).

6.512.2.6 virtual void decaf::internal::io::StandardInputStream::notify () throw (lang::Exception) [inline, protected, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.512.2.7 virtual void decaf::internal::io::StandardInputStream::notifyAll () throw (lang::Exception) [inline, protected, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.512.2.8 virtual int decaf::internal::io::StandardInputStream::read (unsigned char * *buffer*, std::size_t *offset*, std::size_t *bufferSize*) throw (decaf::io::IOException, decaf::lang::exceptions::NullPointerException) [virtual]

Reads an array of bytes from the buffer.

Parameters:

buffer (out) the target buffer.

offset the position in the buffer to start reading from.

bufferSize the size of the output buffer.

Returns:

The number of bytes read.

Exceptions:

decaf::io::IOException *(*p. ***1477***)* thrown if an error occurs.

decaf::lang::exceptions::NullPointerException *(*p. ***1951***)* if buffer is null.

Implements **decaf::io::InputStream** (p. 1408).

6.512.2.9 virtual unsigned char decaf::internal::io::StandardInputStream::read () throw (decaf::io::IOException) [virtual]

Reads a single byte from the buffer.

Returns:

The next byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::InputStream** (p. 1408).

6.512.2.10 virtual void decaf::internal::io::StandardInputStream::reset () throw (decaf::io::IOException) [virtual]

Repositions this stream to the position at the time the mark method was last called on this input stream. If the method markSupported returns true, then: * If the method mark has not been called since the stream was created, or the number of bytes read from the stream since mark was last called is larger than the argument to mark at that last call, then an IOException might be thrown. * If such an IOException is not thrown, then the stream is reset to a state such that all the bytes read since the most recent call to mark (or since the start of the file, if mark has not been called) will be resupplied to subsequent callers of the read method, followed by any bytes that otherwise would have been the next input data as of the time of the call to reset. If the method markSupported returns false, then: * The call to reset may throw an IOException. * If an IOException is not thrown, then the stream is reset to a fixed state that depends on the particular type of the input stream and how it was created. The bytes that will be supplied to subsequent callers of the read method depend on the particular type of the input stream.

Exceptions:

IOException

Implements **decaf::io::InputStream** (p. 1408).

6.512.2.11 virtual std::size_t decaf::internal::io::StandardInputStream::skip (std::size_t *num*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [virtual]

Skips over and discards n bytes of data from this input stream. The skip method may, for a variety of reasons, end up skipping over some smaller number of bytes, possibly 0. This may result from any of a number of conditions; reaching end of file before n bytes have been skipped is only one possibility. The actual number of bytes skipped is returned. If n is negative, no bytes are skipped.

The skip method of InputStream creates a byte array and then repeatedly reads into it until n bytes have been read or the end of the stream has been reached. Subclasses are encouraged to provide a more efficient implementation of this method.

Parameters:

num - the number of bytes to skip

Returns:

total bytes skipped

Exceptions:

decaf::io::IOException *(*p. *1477)* if an error occurs

decaf::lang::exceptions::UnsupportedOperationException *(*p. *2635)* If skip is not supported.

Implements **decaf::io::InputStream** (p. 1409).

6.512.2.12 virtual void decaf::internal::io::StandardInputStream::unlock () throw (lang::Exception) [inline, protected, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.512.2.13 virtual void decaf::internal::io::StandardInputStream::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, protected, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.512.2.14 virtual void decaf::internal::io::StandardInputStream::wait () throw (lang::Exception) [inline, protected, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardInputStream.h**

6.513 decaf::internal::io::StandardOutputStream Class Reference

`#include <src/main/decaf/internal/io/StandardOutputStream.h>`Inheritance diagram for decaf::internal::io::StandardOutputStream:

Public Member Functions

- **StandardOutputStream** ()
- virtual ~**StandardOutputStream** ()
- virtual void **write** (unsigned char c) throw (decaf::io::IOException)

  Writes a single byte to the output stream.

- virtual void **write** (const std::vector< unsigned char > &buffer) throw (decaf::io::IOException)

  Writes an array of bytes to the output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t offset, std::size_t len) throw (decaf::io::IOException, lang::exceptions::NullPointerException)

  Writes an array of bytes to the output stream.

- virtual void **flush** () throw (decaf::io::IOException)

  Invokes flush on the target output stream.

- virtual void **close** () throw (decaf::lang::Exception)

  Invokes close on the target output stream.

- virtual void **lock** () throw (decaf::lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (decaf::lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (decaf::lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (decaf::lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (decaf::lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (decaf::lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

6.513.1 Constructor & Destructor Documentation

6.513.1.1 decaf::internal::io::StandardOutputStream::StandardOutputStream ()

6.513.1.2 virtual decaf::internal::io::StandardOutputStream::~StandardOutputStream () [virtual]

6.513.2 Member Function Documentation

6.513.2.1 virtual void decaf::internal::io::StandardOutputStream::close () throw (decaf::lang::Exception) [inline, virtual]

Invokes close on the target output stream. throws CMSException if an error occurs

Implements **decaf::io::Closeable** (p. 840).

6.513.2.2 virtual void decaf::internal::io::StandardOutputStream::flush () throw (decaf::io::IOException) [virtual]

Invokes flush on the target output stream. throws **decaf::io::IOException** (p. 1477) if an error occurs

Implements **decaf::io::OutputStream** (p. 1992).

6.513.2.3 virtual void decaf::internal::io::StandardOutputStream::lock () throw (decaf::lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.513.2.4 virtual void decaf::internal::io::StandardOutputStream::notify () throw (decaf::lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.513.2.5 virtual void decaf::internal::io::StandardOutputStream::notifyAll () throw (decaf::lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.513.2.6 virtual void decaf::internal::io::StandardOutputStream::unlock () throw (decaf::lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.513.2.7 virtual void decaf::internal::io::StandardOutputStream::wait (unsigned long *millisecs*) throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.513.2.8 virtual void decaf::internal::io::StandardOutputStream::wait () throw (decaf::lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

6.513.2.9 virtual void decaf::internal::io::StandardOutputStream::write (const unsigned char * *buffer*, std::size_t *offset*, std::size_t *len*) throw (decaf::io::IOException, lang::exceptions::NullPointerException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The array of bytes to write.

offset,the position to start writing in buffer.

len The number of bytes from the buffer to be written.

Exceptions:

decaf::io::IOException *(*p. ***1477)*** thrown if an error occurs.

NullPointerException if buffer is null.

Implements **decaf::io::OutputStream** (p. 1993).

6.513.2.10 virtual void decaf::internal::io::StandardOutputStream::write (const std::vector< unsigned char > & *buffer*) throw (decaf::io::IOException) [virtual]

Writes an array of bytes to the output stream.

Parameters:

buffer The bytes to write.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

6.513.2.11 virtual void decaf::internal::io::StandardOutputStream::write (unsigned char *c*) throw (decaf::io::IOException) [virtual]

Writes a single byte to the output stream.

Parameters:

c the byte.

Exceptions:

IOException thrown if an error occurs.

Implements **decaf::io::OutputStream** (p. 1993).

The documentation for this class was generated from the following file:

- src/main/decaf/internal/io/**StandardOutputStream.h**

6.514 cms::Startable Class Reference

Interface for a class that implements the start method.

`#include <src/main/cms/Startable.h>`Inheritance diagram for cms::Startable:

Public Member Functions

- virtual ~**Startable** ()
- virtual void **start** ()=0 throw (CMSException)

  Starts the service.

6.514.1 Detailed Description

Interface for a class that implements the start method. An object that implements the **Startable** (p. 2413) interface implies that until its start method is called it will be considered to be in a closed or stopped state and will throw an Exception to indicate that it is not in an started state if one of its methods is called.

Since:

1.0

6.514.2 Constructor & Destructor Documentation

6.514.2.1 virtual cms::Startable::~Startable () [inline, virtual]

6.514.3 Member Function Documentation

6.514.3.1 virtual void cms::Startable::start () throw (CMSException) [pure virtual]

Starts the service.

Exceptions:

CMSException *(*p. ***850****)* if an internal error occurs while starting.

Implemented in **activemq::core::ActiveMQConnection** (p. 198), **activemq::transport::correlator::ResponseCorrelator** (p. 2240), **activemq::transport::failover::FailoverTransport** (p. 1307), **activemq::transport::IOTransport** (p. 1485), **activemq::transport::mock::MockTransport** (p. 1917), **activemq::transport::TransportFilter** (p. 2622), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1987).

The documentation for this class was generated from the following file:

- src/main/cms/**Startable.h**

6.515 decaf::lang::STATIC_CAST_TOKEN Struct Reference

`#include <src/main/decaf/lang/Pointer.h>`

The documentation for this struct was generated from the following file:

- src/main/decaf/lang/**Pointer.h**

6.516 activemq::core::ActiveMQConstants::StaticInitializer Class Reference

`#include <src/main/activemq/core/ActiveMQConstants.h>`

Public Member Functions

- **StaticInitializer** ()
- virtual ~**StaticInitializer** ()

Static Public Attributes

- static std::string **destOptions** [NUM_OPTIONS]
- static std::string **uriParams** [NUM_PARAMS]
- static std::map< std::string, **DestinationOption** > **destOptionMap**
- static std::map< std::string, **URIParam** > **uriParamsMap**

6.516.1 Constructor & Destructor Documentation

6.516.1.1 activemq::core::ActiveMQConstants::StaticInitializer::StaticInitializer ()

6.516.1.2 virtual activemq::core::ActiveMQConstants::StaticInitializer::~StaticInitializer () `[inline, virtual]`

6.516.2 Field Documentation

6.516.2.1 std::map<std::string, DestinationOption> activemq::core::ActiveMQConstants::StaticInitializer::destOptionMap `[static]`

6.516.2.2 std::string activemq::core::ActiveMQConstants::StaticInitializer::destOptions[NUM_OPTIONS] `[static]`

6.516.2.3 std::string activemq::core::ActiveMQConstants::StaticInitializer::uriParams[NUM_PARAMS] `[static]`

6.516.2.4 std::map<std::string, URIParam> activemq::core::ActiveMQConstants::StaticInitializer::uriParamsMap `[static]`

The documentation for this class was generated from the following file:

- src/main/activemq/core/**ActiveMQConstants.h**

6.517 decaf::util::StlList< E > Class Template Reference

List (p. 1591) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

`#include <src/main/decaf/util/StlList.h>`Inheritance diagram for decaf::util::StlList< E >:

Data Structures

- class **ConstStlListIterator**
- class **StlListIterator**

Public Member Functions

- **StlList** ()

  Default constructor - does nothing.

- **StlList** (const **StlList** &source)

  Copy constructor - copies the content of the given set into this one.

- **StlList** (const **Collection**< E > &source)

  Copy constructor - copies the content of the given set into this one.

- virtual ~**StlList** ()
- virtual bool **equals** (const **StlList** &source) const

  Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.
  Returns:

  true if the Collections contain the same elements.

- virtual **Iterator**< E > * **iterator** ()

  Returns:

  an iterator over a set of elements of type T.

- virtual **Iterator**< E > * **iterator** () const
- virtual **ListIterator**< E > * **listIterator** ()

  Returns:

  a list iterator over the elements in this list (in proper sequence).

- virtual **ListIterator**< E > * **listIterator** () const
- virtual **ListIterator**< E > * **listIterator** (std::size_t index) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Parameters:

  index *index of first element to be returned from the list iterator (by a call to the next method).*

Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException *if the index is out of range (index < 0 || index > **size()** (p. 878))*

- virtual **ListIterator**< E > * **listIterator** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)
- virtual void **copy** (const **StlList** &source)

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

Removes all of the elements from this collection (optional operation).
The collection will be empty after this method returns.
*This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1490) operation. Most implementations will probably choose to override this method for efficiency.*
Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException *if the clear operation is not supported by this collection*

- virtual bool **contains** (const E &value) const throw (lang::Exception)

Returns true if this collection contains the specified element.
This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value *- the value whose presence is to be queried for in this **Collection** (p. 871).*

Returns:

true if the value is contained in this collection

Exceptions:

Exception *if an error occurs,*

- virtual std::size_t **indexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.
More formally, returns the lowest index i such that get(i) == value, or -1 if there is no such index.

Parameters:

value *- element to search for*

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException *if value is not in the list*

- virtual std::size_t **lastIndexOf** (const E &value) throw (decaf::lang::exceptions::NoSuchElementException)

  Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.
  More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.
  Parameters:

  value *- element to search for*
  Returns:

  the index of the last occurrence of the specified element in this list.
  Exceptions:

  NoSuchElementException *if value is not in the list*

- virtual bool **isEmpty** () const

  Returns true if this collection contains no elements.
  This implementation returns ***size()*** *(p. 878) == 0.*
  Returns:

  true if the size method return 0.

- virtual std::size_t **size** () const

  Returns the number of elements in this collection.
  If this collection contains more than Integer.MAX_ VALUE elements, returns Integer.MAX_ - VALUE.
  Returns:

  the number of elements in this collection

- virtual E **get** (std::size_t index) const throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Gets the element contained at position passed.
  Parameters:

  index *- position to get*
  Returns:

  value at index

- virtual E **set** (std::size_t index, const E &element) throw (decaf::lang::exceptions::IndexOutOfBoundsException)

  Replaces the element at the specified position in this list with the specified element.
  Parameters:

  index *- index of the element to replace*
  element *- element to be stored at the specified position*
  Returns:

  the element previously at the specified position
  Exceptions:

  IndexOutOfBoundsException *- if the index is greater than size*

- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

Returns true if this collection changed as a result of the call.
(Returns false if this collection does not permit duplicates and already contains the specified element.)
Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. ***Collection*** *(p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.*
If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.
For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value *- reference to the element to add.*

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException
IllegalArgumentException
IllegalStateException *if the element cannot be added at this time due to insertion restrictions*

- virtual void **add** (std::size_t index, const E &element) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException)

Inserts the specified element at the specified position in this list.
Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index *- index at which the specified element is to be inserted*
element *- element to be inserted*

Exceptions:

IndexOutOfBoundsException *- if the index is greater than size*
UnsupportedOperationException *- If the collection is non-modifiable.*

- virtual bool **addAll** (std::size_t index, const **Collection**< E > &source) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).
Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index *The index at which to insert the first element from the specified collection*
source *The* ***Collection*** *(p. 871) containing elements to be added to this list*

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException *- if the index is greater than size*
UnsupportedOperationException *- If the collection is non-modifiable.*

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes a single instance of the specified element from this collection, if it is present (optional operation).
  More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).
  This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.
  Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

  Parameters:

  value *- element to be removed from this collection, if present*

  Returns:

  true if an element was removed as a result of this call

  Exceptions:

  UnsupportedOperationException *if the remove operation is not supported by this collection.*
  IllegalArgumentException *If the value is not a valid entry for this* ***Collection*** *(p. 871).*

- virtual E **remove** (std::size_t index) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException)

  Removes the element at the specified position in this list.
  Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

  Parameters:

  index *- the index of the element to be removed*

  Returns:

  the element previously at the specified position

  Exceptions:

  IndexOutOfBoundsException *- if the index is greater than size*
  UnsupportedOperationException *- If the collection is non-modifiable.*

6.517.1 Detailed Description

template<typename E> class decaf::util::StlList< E >

List (p. 1591) class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.

6.517.2 Constructor & Destructor Documentation

6.517.2.1 template<typename E> decaf::util::StlList< E >::StlList () [inline]

Default constructor - does nothing.

6.517.2.2 template<typename E> decaf::util::StlList< E >::StlList (const StlList< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.517.2.3 template<typename E> decaf::util::StlList< E >::StlList (const Collection< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.517.2.4 template<typename E> virtual decaf::util::StlList< E >::~StlList () [inline, virtual]

6.517.3 Member Function Documentation

6.517.3.1 template<typename E> virtual void decaf::util::StlList< E >::add (std::size_t *index*, const E & *element*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Inserts the specified element at the specified position in this list.

Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).

Parameters:

index - index at which the specified element is to be inserted

element - element to be inserted

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 1592).

6.517.3.2 template<typename E> virtual bool decaf::util::StlList< E >::add (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 873).

Referenced by decaf::util::StlList< Pointer< BackupTransport > >::addAll().

6.517.3.3 template<typename E> virtual bool decaf::util::StlList< E >::addAll (std::size_t *index*, const Collection< E > & *source*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Inserts all of the elements in the specified collection into this list at the specified position (optional operation).

Shifts the element currently at that position (if any) and any subsequent elements to the right (increases their indices). The new elements will appear in this list in the order that they are returned by the specified collection's iterator. The behavior of this operation is undefined if the specified collection is modified while the operation is in progress. (Note that this will occur if the specified collection is this list, and it's nonempty.)

Parameters:

index The index at which to insert the first element from the specified collection

source The **Collection** (p. 871) containing elements to be added to this list

Returns:

true if this list changed as a result of the call

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 1592).

6.517.3.4 template<typename E> virtual void decaf::util::StlList< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1490) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 127).

6.517.3.5 template<typename E> virtual bool decaf::util::StlList< E >::contains (const E & *value*) const throw (lang::Exception) [inline, virtual]

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 871).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 127).

6.517.3.6 template<typename E> virtual void decaf::util::StlList< E >::copy (const StlList< E > & *source*) [inline, virtual]

Referenced by decaf::util::StlList< Pointer< BackupTransport > >::StlList().

6.517.3.7 template<typename E> virtual bool decaf::util::StlList< E >::equals (const StlList< E > & *source*) const [inline, virtual]

Compares the passed collection to this one, if they contain the same elements, i.e.

all their elements are equivalent, then it returns true.

Returns:

true if the Collections contain the same elements.

Implements **decaf::util::Collection< E >** (p. 875).

6.517.3.8 template<typename E> virtual E decaf::util::StlList< E >::get (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Gets the element contained at position passed.

Parameters:

index - position to get

Returns:

value at index

Implements **decaf::util::List< E >** (p. 1593).

6.517.3.9 template<typename E> virtual std::size_t decaf::util::StlList< E >::indexOf (const E & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]

Returns the index of the first occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the lowest index i such that get(i) == value, or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the first occurrence of the specified element in this list,

Exceptions:

NoSuchElementException if value is not in the list

Implements **decaf::util::List< E >** (p. 1593).

6.517.3.10 template<typename E> virtual bool decaf::util::StlList< E >::isEmpty () const [inline, virtual]

Returns true if this collection contains no elements.

This implementation returns **size()** (p. 878) == 0.

Returns:

true if the size method return 0.

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 129).

6.517.3.11 template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1487).

6.517.3.12 template<typename E> virtual Iterator<E>* decaf::util::StlList< E >::iterator () [inline, virtual]

Returns:

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1487).

6.517.3.13 template<typename E> virtual std::size_t decaf::util::StlList< E >::lastIndexOf (const E & *value*) throw (decaf::lang::exceptions::NoSuchElementException) [inline, virtual]

Returns the index of the last occurrence of the specified element in this list, or -1 if this list does not contain the element.

More formally, returns the highest index i such that get(i) == value or -1 if there is no such index.

Parameters:

value - element to search for

Returns:

the index of the last occurrence of the specified element in this list.

Exceptions:

NoSuchElementException if value is not in the list

Implements **decaf::util::List< E >** (p. 1594).

6.517.3.14 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t *index*) const throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Implements **decaf::util::List< E >** (p. 1594).

6.517.3.15 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator (std::size_t *index*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Parameters:

index index of first element to be returned from the list iterator (by a call to the next method).

Returns:

a list iterator of the elements in this list (in proper sequence), starting at the specified position in this list. The specified index indicates the first element that would be returned by an initial call to next. An initial call to previous would return the element with the specified index minus one.

Exceptions:

IndexOutOfBoundsException if the index is out of range (index < 0 || index > **size()** (p. 878))

Implements **decaf::util::List< E >** (p. 1595).

6.517.3.16 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () const [inline, virtual]

Implements **decaf::util::List< E >** (p. 1595).

6.517.3.17 template<typename E> virtual ListIterator<E>* decaf::util::StlList< E >::listIterator () [inline, virtual]

Returns:

a list iterator over the elements in this list (in proper sequence).

Implements **decaf::util::List< E >** (p. 1595).

6.517.3.18 template<typename E> virtual E decaf::util::StlList< E >::remove (std::size_t *index*) throw (decaf::lang::exceptions::UnsupportedOperationException, decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Removes the element at the specified position in this list.

Shifts any subsequent elements to the left (subtracts one from their indices). Returns the element that was removed from the list.

Parameters:

index - the index of the element to be removed

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

UnsupportedOperationException - If the collection is non-modifiable.

Implements **decaf::util::List< E >** (p. 1596).

6.517.3.19 template<typename E> virtual bool decaf::util::StlList< E >::remove (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 871).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 130).

6.517.3.20 template<typename E> virtual E decaf::util::StlList< E >::set (std::size_t *index*, const E & *element*) throw (decaf::lang::exceptions::IndexOutOfBoundsException) [inline, virtual]

Replaces the element at the specified position in this list with the specified element.

Parameters:

index - index of the element to replace

element - element to be stored at the specified position

Returns:

the element previously at the specified position

Exceptions:

IndexOutOfBoundsException - if the index is greater than size

Implements **decaf::util::List< E >** (p. 1596).

6.517.3.21 template<typename E> virtual std::size_t decaf::util::StlList< E >::size () const [inline, virtual]

Returns the number of elements in this collection.

If this collection contains more than Integer.MAX_VALUE elements, returns Integer.MAX_VALUE.

Returns:

the number of elements in this collection

Implements **decaf::util::Collection< E >** (p. 878).

Referenced by decaf::util::StlList< Pointer< BackupTransport > >::add(), decaf::util::StlList< Pointer< BackupTransport > >::addAll(), decaf::util::StlList< Pointer< BackupTransport > >::get(), decaf::util::StlList< Pointer< BackupTransport > >::lastIndexOf(), decaf::util::StlList< Pointer< BackupTransport > >::listIterator(), decaf::util::StlList< Pointer< BackupTransport > >::remove(), and decaf::util::StlList< Pointer< BackupTransport > >::set().

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlList.h**

6.518 decaf::util::StlMap< K, V, COMPARATOR > Class Template Reference

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

`#include <src/main/decaf/util/StlMap.h>`Inheritance diagram for decaf::util::StlMap< K, V, COMPARATOR >:

Public Member Functions

- **StlMap** ()

  Default constructor - does nothing.

- **StlMap** (const **StlMap** &source)

  Copy constructor - copies the content of the given map into this one.

- **StlMap** (const **Map**< K, V, COMPARATOR > &source)

  Copy constructor - copies the content of the given map into this one.

- virtual ~**StlMap** ()
- virtual bool **equals** (const **StlMap** &source) const

  Comparison, equality is dependent on the method of determining if the element are equal.
  Parameters:

  source *- **Map** (p. 1689) to compare to this one.*
  Returns:

  *true if the **Map** (p. 1689) passed is equal in value to this one.*

- virtual bool **equals** (const **Map**< K, V, COMPARATOR > &source) const

- virtual void **copy** (const **StlMap** &source)

  Copies the content of the source map into this map.
  Erases all existing data in this map.
  Parameters:

  source *The source object to copy from.*

- virtual void **copy** (const **Map**< K, V, COMPARATOR > &source)

- virtual void **clear** () throw (decaf::lang::exceptions::UnsupportedOperationException)

  Removes all keys and values from this map.
  Exceptions:

  UnsupportedOperationException *if this map is unmodifiable.*

- virtual bool **containsKey** (const K &key) const

  Indicates whether or this map contains a value for the given key.

Parameters:

key *The key to look up.*

Returns:

true if this map contains the value, otherwise false.

- virtual bool **containsValue** (const V &value) const

  Indicates whether or this map contains a value for the given value, i.e.
  they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

  Parameters:

  value *The Value to look up.*

  Returns:

  true if this map contains the value, otherwise false.

- virtual bool **isEmpty** () const

  Returns:

  *if the **Map** (p. 1689) contains any element or not, TRUE or FALSE*

- virtual std::size_t **size** () const

  Returns:

  The number of elements (key/value pairs) in this map.

- virtual V & **get** (const K &key) throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*
  If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

  Parameters:

  key *The search key.*

  Returns:

  A reference to the value for the given key.

  Exceptions:

  NoSuchElementException *if the key requests doesn't exist in the **Map** (p. 1689).*

- virtual const V & **get** (const K &key) const throw (lang::exceptions::NoSuchElementException)

  *Gets the value mapped to the specified key in the **Map** (p. 1689).*
  If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

  Parameters:

  key *The search key.*

  Returns:

  A {const} reference to the value for the given key.

  Exceptions:

  NoSuchElementException *if the key requests doesn't exist in the **Map** (p. 1689).*

- virtual void **put** (const K &key, const V &value) throw (decaf::lang::exceptions::UnsupportedOperationException)

  Sets the value for the specified key.

Parameters:

key *The target key.*
value *The value to be set.*

Exceptions:

UnsupportedOperationException *if this map is unmodifiable.*

- virtual void **putAll** (const **StlMap**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

  *Stores a copy of the Mappings contained in the other **Map*** (p. *1689*) *in this one.*

  Parameters:

  other *A **Map*** (p. *1689*) *instance whose elements are to all be inserted in this **Map*** (p. *1689*).

  Exceptions:

  UnsupportedOperationException *If the implementing class does not support the putAll operation.*

- virtual void **putAll** (const **Map**< K, V, COMPARATOR > &other) throw (decaf::lang::exceptions::UnsupportedOperationException)

- virtual V **remove** (const K &key) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException)

  Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

  Parameters:

  key *The search key.*

  Returns:

  a copy of the element that was previously mapped to the given key

  Exceptions:

  NoSuchElementException *if this key is not in the **Map*** (p. *1689*).
  UnsupportedOperationException *if this map is unmodifiable.*

- virtual std::vector< K > **keySet** () const

  *Returns a **Set*** (p. *2320*) *view of the mappings contained in this map.*
  *The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove*** (p. *1490*), ***Set.remove*** (p. *130*), *removeAll, retainAll and clear operations. It does not support the add or addAll operations.*

  Returns:

  the entire set of keys in this map as a std::vector.

- virtual std::vector< V > **values** () const

  Returns:

  the entire set of values in this map as a std::vector.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)
  Unlocks the object.
- virtual void **wait** () throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** () throw (lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** () throw (lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.518.1 Detailed Description

template<typename K, typename V, typename COMPARATOR = std::less<K>> class decaf::util::StlMap< K, V, COMPARATOR >

Map (p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.

Since:

1.0

6.518.2 Constructor & Destructor Documentation

6.518.2.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap () [inline]

Default constructor - does nothing.

6.518.2.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const StlMap< K, V, COMPARATOR > & *source*) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.518.2.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> decaf::util::StlMap< K, V, COMPARATOR >::StlMap (const Map< K, V, COMPARATOR > & *source*) [inline]

Copy constructor - copies the content of the given map into this one.

Parameters:

source The source map.

6.518.2.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual decaf::util::StlMap< K, V, COMPARATOR >::~StlMap () [inline, virtual]

6.518.3 Member Function Documentation

6.518.3.1 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::clear () throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all keys and values from this map.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1690).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::copy().

6.518.3.2 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsKey (const K & *key*) const [inline, virtual]

Indicates whether or this map contains a value for the given key.

Parameters:

key The key to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1691).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::equals().

6.518.3.3 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::containsValue (const V & *value*) const [inline, virtual]

Indicates whether or this map contains a value for the given value, i.e.

they are equal, this is done by operator== so the types must pass equivalence testing in this manner.

Parameters:

value The Value to look up.

Returns:

true if this map contains the value, otherwise false.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1692).

6.518.3.4 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy (const Map< K, V, COMPARATOR > & *source*) [inline, virtual]

6.518.3.5 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::copy (const StlMap< K, V, COMPARATOR > & *source*) [inline, virtual]

Copies the content of the source map into this map.

Erases all existing data in this map.

Parameters:

source The source object to copy from.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::StlMap().

6.518.3.6 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const Map< K, V, COMPARATOR > & *source*) const [inline, virtual]

6.518.3.7 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::equals (const StlMap< K, V, COMPARATOR > & *source*) const [inline, virtual]

Comparison, equality is dependent on the method of determining if the element are equal.

Parameters:

source - **Map** (p. 1689) to compare to this one.

Returns:

true if the **Map** (p. 1689) passed is equal in value to this one.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

6.518.3.8 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual const V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & *key*) const throw (lang::exceptions::NoSuchElementException) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689).

If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A {const} reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1693).

6.518.3.9 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V& decaf::util::StlMap< K, V, COMPARATOR >::get (const K & *key*) throw (lang::exceptions::NoSuchElementException) [inline, virtual]

Gets the value mapped to the specified key in the **Map** (p. 1689).

If there is no element in the map whose key is equivalent to the key provided then a NoSuchElementException is thrown.

Parameters:

key The search key.

Returns:

A reference to the value for the given key.

Exceptions:

NoSuchElementException if the key requests doesn't exist in the **Map** (p. 1689).

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1694).

6.518.3.10 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual bool decaf::util::StlMap< K, V, COMPARATOR >::isEmpty () const [inline, virtual]

Returns:

if the **Map** (p. 1689) contains any element or not, TRUE or FALSE

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1695).

6.518.3.11 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<K> decaf::util::StlMap< K, V, COMPARATOR >::keySet () const [inline, virtual]

Returns a **Set** (p. 2320) view of the mappings contained in this map.

The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress (except through the iterator's own remove operation, or through the setValue operation on a map entry returned by the iterator) the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the **Iterator.remove** (p. 1490), **Set.remove** (p. 130), removeAll, retainAll and clear operations. It does not support the add or addAll operations.

Returns:

the entire set of keys in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1696).

6.518.3.12 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.518.3.13 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.518.3.14 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.518.3.15 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::put (const K & *key*, const V & *value*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Sets the value for the specified key.

Parameters:

key The target key.

value The value to be set.

Exceptions:

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1697).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::putAll().

6.518.3.16 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const Map< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

6.518.3.17 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::putAll (const StlMap< K, V, COMPARATOR > & *other*) throw (decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Stores a copy of the Mappings contained in the other **Map** (p. 1689) in this one.

Parameters:

other A **Map** (p. 1689) instance whose elements are to all be inserted in this **Map** (p. 1689).

Exceptions:

UnsupportedOperationException If the implementing class does not support the putAll operation.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1697).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::copy().

6.518.3.18 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual V decaf::util::StlMap< K, V, COMPARATOR >::remove (const K & *key*) throw (decaf::lang::exceptions::NoSuchElementException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes the value (key/value pair) for the specified key from the map, returns a copy of the value that was mapped to the key.

Parameters:

key The search key.

Returns:

a copy of the element that was previously mapped to the given key

Exceptions:

NoSuchElementException if this key is not in the **Map** (p. 1689).

UnsupportedOperationException if this map is unmodifiable.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1698).

6.518.3.19 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::size_t decaf::util::StlMap< K, V, COMPARATOR >::size () const [inline, virtual]

Returns:

The number of elements (key/value pairs) in this map.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1699).

6.518.3.20 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.518.3.21 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual std::vector<V> decaf::util::StlMap< K, V, COMPARATOR >::values () const [inline, virtual]

Returns:

the entire set of values in this map as a std::vector.

Implements **decaf::util::Map< K, V, COMPARATOR >** (p. 1700).

Referenced by decaf::util::StlMap< std::string, cms::Topic * >::values().

6.518.3.22 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.518.3.23 template<typename K, typename V, typename COMPARATOR = std::less<K>> virtual void decaf::util::StlMap< K, V, COMPARATOR >::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlMap.h**

6.519 decaf::util::StlQueue< T > Class Template Reference

The **Queue** (p. 2154) class accepts messages with an psuh(m) command where m is the message to be queued.

`#include <src/main/decaf/util/StlQueue.h>`Inheritance diagram for decaf::util::StlQueue< T >:

Data Structures

- class **QueueIterator**

Public Member Functions

- **StlQueue** ()
- virtual ~**StlQueue** ()
- **Iterator**< T > * **iterator** ()

  *Gets an **Iterator** (p. 1489) over this **Queue** (p. 2154).*

- void **clear** ()

  Empties this queue.

- T & **front** ()

  Returns a Reference to the element at the head of the queue.

- const T & **front** () const

  Returns a Reference to the element at the head of the queue.

- T & **back** ()

  Returns a Reference to the element at the tail of the queue.

- const T & **back** () const

  Returns a Reference to the element at the tail of the queue.

- void **push** (const T &t)

  Places a new Object at the Tail of the queue.

- void **enqueueFront** (const T &t)

  Places a new Object at the front of the queue.

- T **pop** ()

  Removes and returns the element that is at the Head of the queue.

- size_t **size** () const

  *Gets the Number of elements currently in the **Queue** (p. 2154).*

- bool **empty** () const

  *Checks if this **Queue** (p. 2154) is currently empty.*

- virtual std::vector< T > **toArray** () const
- void **reverse** (**StlQueue**< T > &target) const

  Reverses the order of the contents of this queue and stores them in the target queue.

- virtual void **lock** () throw (lang::Exception)

  Locks the object.

- virtual void **unlock** () throw (lang::Exception)

  Unlocks the object.

- virtual void **wait** () throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **wait** (unsigned long millisecs) throw (lang::Exception)

  Waits on a signal from this object, which is generated by a call to Notify.

- virtual void **notify** () throw (lang::Exception)

  Signals a waiter on this object that it can now wake up and continue.

- virtual void **notifyAll** () throw (lang::Exception)

  Signals the waiters on this object that it can now wake up and continue.

- T & **getSafeValue** ()

  Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

6.519.1 Detailed Description

template<typename T> class decaf::util::StlQueue< T >

The **Queue** (p. 2154) class accepts messages with an psuh(m) command where m is the message to be queued. It destructively returns the message with **pop()** (p. 2444). **pop()** (p. 2444) returns messages in the order they were enqueued.

Queue (p. 2154) is implemented with an instance of the STL queue object. The interface is essentially the same as that of the STL queue except that the pop method actually reaturns a reference to the element popped. This frees the app from having to call the `front` method before calling pop.

Queue<string> sq; // make a queue to hold string messages sq.push(s); // enqueues a message m string s = sq.pop(); // dequeues a message

= DESIGN CONSIDERATIONS

The **Queue** (p. 2154) class inherits from the Synchronizable interface and provides methods for locking and unlocking this queue as well as waiting on this queue. In a multi-threaded app this can allow for multiple threads to be reading from and writing to the same **Queue** (p. 2154).

Clients should consider that in a multiple threaded app it is possible that items could be placed on the queue faster than you are taking them off, so protection should be placed in your polling loop to ensure that you don't get stuck there.

6.519.2 Constructor & Destructor Documentation

6.519.2.1 template<typename T> decaf::util::StlQueue< T >::StlQueue () [inline]

6.519.2.2 template<typename T> virtual decaf::util::StlQueue< T >::~StlQueue () [inline, virtual]

6.519.3 Member Function Documentation

6.519.3.1 template<typename T> const T& decaf::util::StlQueue< T >::back () const [inline]

Returns a Reference to the element at the tail of the queue.

Returns:

reference to a queue type object or (safe)

6.519.3.2 template<typename T> T& decaf::util::StlQueue< T >::back () [inline]

Returns a Reference to the element at the tail of the queue.

Returns:

reference to a queue type object or (safe)

6.519.3.3 template<typename T> void decaf::util::StlQueue< T >::clear () [inline]

Empties this queue.

6.519.3.4 template<typename T> bool decaf::util::StlQueue< T >::empty () const [inline]

Checks if this **Queue** (p. 2154) is currently empty.

Returns:

boolean indicating queue emptiness

6.519.3.5 template<typename T> void decaf::util::StlQueue< T >::enqueueFront (const T & *t*) [inline]

Places a new Object at the front of the queue.

Parameters:

t - **Queue** (p. 2154) Object Type reference.

6.519.3.6 template<typename T> const T& decaf::util::StlQueue< T >::front () const [inline]

Returns a Reference to the element at the head of the queue.

Returns:

reference to a queue type object or (safe)

6.519.3.7 template<typename T> T& decaf::util::StlQueue< T >::front () [inline]

Returns a Reference to the element at the head of the queue.

Returns:

reference to a queue type object or (safe)

6.519.3.8 template<typename T> T& decaf::util::StlQueue< T >::getSafeValue () [inline]

Fetch a reference to the safe value this object will return when there is nothing to fetch from the queue.

Returns:

Reference to this Queues safe object

Referenced by decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::back(), decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::front(), and decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >::pop().

6.519.3.9 template<typename T> Iterator<T>* decaf::util::StlQueue< T >::iterator () [inline]

Gets an **Iterator** (p. 1489) over this **Queue** (p. 2154).

Returns:

new iterator pointer that is owned by the caller.

6.519.3.10 template<typename T> virtual void decaf::util::StlQueue< T >::lock () throw (lang::Exception) [inline, virtual]

Locks the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2508).

6.519.3.11 template<typename T> virtual void decaf::util::StlQueue< T >::notify () throw (lang::Exception) [inline, virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Implements **decaf::util::concurrent::Synchronizable** (p. 2510).

6.519.3.12 template<typename T> virtual void decaf::util::StlQueue< T >::notifyAll () throw (lang::Exception) [inline, virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Implements **decaf::util::concurrent::Synchronizable** (p. 2511).

6.519.3.13 template<typename T> T decaf::util::StlQueue< T >::pop () [inline]

Removes and returns the element that is at the Head of the queue.

Returns:

reference to a queue type object or (safe)

6.519.3.14 template<typename T> void decaf::util::StlQueue< T >::push (const T & *t*) [inline]

Places a new Object at the Tail of the queue.

Parameters:

t - **Queue** (p. 2154) Object Type reference.

6.519.3.15 template<typename T> void decaf::util::StlQueue< T >::reverse (StlQueue< T > & *target*) const [inline]

Reverses the order of the contents of this queue and stores them in the target queue.

Parameters:

target - The target queue that will receive the contents of this queue in reverse order.

6.519.3.16 template<typename T> size_t decaf::util::StlQueue< T >::size () const [inline]

Gets the Number of elements currently in the **Queue** (p. 2154).

Returns:

Queue (p. 2154) Size

6.519.3.17 template<typename T> virtual std::vector<T> decaf::util::StlQueue< T >::toArray () const [inline, virtual]

Returns:

the all values in this queue as a std::vector.

6.519.3.18 template<typename T> virtual void decaf::util::StlQueue< T >::unlock () throw (lang::Exception) [inline, virtual]

Unlocks the object.

Implements **decaf::util::concurrent::Synchronizable** (p. 2512).

6.519.3.19 template<typename T> virtual void decaf::util::StlQueue< T >::wait (unsigned long *millisecs*) throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs time to wait, or WAIT_INIFINITE

Exceptions:

ActiveMQException

Implements **decaf::util::concurrent::Synchronizable** (p. 2513).

6.519.3.20 template<typename T> virtual void decaf::util::StlQueue< T >::wait () throw (lang::Exception) [inline, virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Implements **decaf::util::concurrent::Synchronizable** (p. 2514).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlQueue.h**

6.520 decaf::util::StlSet< E > Class Template Reference

Set (p. 2320) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

`#include <src/main/decaf/util/StlSet.h>`Inheritance diagram for decaf::util::StlSet< E >:

Data Structures

- class **ConstSetIterator**
- class **SetIterator**

Public Member Functions

- **StlSet** ()

  Default constructor - does nothing.

- **StlSet** (const **StlSet** &source)

  Copy constructor - copies the content of the given set into this one.

- **StlSet** (const **Collection**< E > &source)

  Copy constructor - copies the content of the given set into this one.

- virtual ~**StlSet** ()
- **Iterator**< E > * **iterator** ()

  Returns:

  an iterator over a set of elements of type T.

- **Iterator**< E > * **iterator** () const
- virtual bool **equals** (const **StlSet** &source) const

  Compares the passed collection to this one, if they contain the same elements, i.e. all their elements are equivalent, then it returns true.
  Returns:

  true if the Collections contain the same elements.

- virtual void **copy** (const **StlSet** &source)

- virtual void **clear** () throw (lang::exceptions::UnsupportedOperationException)

  Removes all of the elements from this collection (optional operation).
  The collection will be empty after this method returns.
  *This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1490) operation. Most implementations will probably choose to override this method for efficiency.*
  Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.
  Exceptions:

  UnsupportedOperationException *if the clear operation is not supported by this collection*

- virtual bool **contains** (const E &value) const throw (lang::Exception)

  Returns true if this collection contains the specified element.
  This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.
  Parameters:

  value *- the value whose presence is to be queried for in this* ***Collection*** *(p. 871).*
  Returns:

  true if the value is contained in this collection
  Exceptions:

  Exception *if an error occurs,*

- virtual bool **isEmpty** () const
- virtual std::size_t **size** () const
- virtual bool **add** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException)

  Returns true if this collection changed as a result of the call.
  (Returns false if this collection does not permit duplicates and already contains the specified element.)
  Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. ***Collection*** *(p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.*
  If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.
  For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.
  Parameters:

  value *- reference to the element to add.*
  Returns:

  true if the element was added
  Exceptions:

  UnsupportedOperationException
  IllegalArgumentException
  IllegalStateException *if the element cannot be added at this time due to insertion restrictions*

- virtual bool **remove** (const E &value) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException)

  Removes a single instance of the specified element from this collection, if it is present (optional operation).
  More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).
  This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.
  Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.
  Parameters:

  value *- element to be removed from this collection, if present*

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException *if the remove operation is not supported by this collection.*

IllegalArgumentException *If the value is not a valid entry for this* ***Collection*** *(p. 871).*

6.520.1 Detailed Description

template<typename E> class decaf::util::StlSet< E >

Set (p. 2320) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.

6.520.2 Constructor & Destructor Documentation

6.520.2.1 template<typename E> decaf::util::StlSet< E >::StlSet () [inline]

Default constructor - does nothing.

6.520.2.2 template<typename E> decaf::util::StlSet< E >::StlSet (const StlSet< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.520.2.3 template<typename E> decaf::util::StlSet< E >::StlSet (const Collection< E > & *source*) [inline]

Copy constructor - copies the content of the given set into this one.

Parameters:

source The source set.

6.520.2.4 template<typename E> virtual decaf::util::StlSet< E >::~StlSet () [inline, virtual]

6.520.3 Member Function Documentation

6.520.3.1 template<typename E> virtual bool decaf::util::StlSet< E >::add (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException, lang::exceptions::IllegalStateException) [inline, virtual]

Returns true if this collection changed as a result of the call.

(Returns false if this collection does not permit duplicates and already contains the specified element.)

Collections that support this operation may place limitations on what elements may be added to this collection. In particular, some collections will refuse to add null elements, and others will impose restrictions on the type of elements that may be added. **Collection** (p. 871) classes should clearly specify in their documentation any restrictions on what elements may be added.

If a collection refuses to add a particular element for any reason other than that it already contains the element, it must throw an exception (rather than returning false). This preserves the invariant that a collection always contains the specified element after this call returns.

For non-pointer values, i.e. class instances or string's the object will be copied into the collection, thus the object must support being copied, must not hide the copy constructor and assignment operator.

Parameters:

value - reference to the element to add.

Returns:

true if the element was added

Exceptions:

UnsupportedOperationException

IllegalArgumentException

IllegalStateException if the element cannot be added at this time due to insertion restrictions

Implements **decaf::util::Collection< E >** (p. 873).

6.520.3.2 template<typename E> virtual void decaf::util::StlSet< E >::clear () throw (lang::exceptions::UnsupportedOperationException) [inline, virtual]

Removes all of the elements from this collection (optional operation).

The collection will be empty after this method returns.

This implementation iterates over this collection, removing each element using the **Iterator.remove** (p. 1490) operation. Most implementations will probably choose to override this method for efficiency.

Note that this implementation will throw an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection is non-empty.

Exceptions:

UnsupportedOperationException if the clear operation is not supported by this collection

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 127).

6.520.3.3 template<typename E> virtual bool decaf::util::StlSet< E >::contains (const E & *value*) const throw (lang::Exception) [inline, virtual]

Returns true if this collection contains the specified element.

This implementation iterates over the elements in the collection, checking each element in turn for equality with the specified element.

Parameters:

value - the value whose presence is to be queried for in this **Collection** (p. 871).

Returns:

true if the value is contained in this collection

Exceptions:

Exception if an error occurs,

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 127).

6.520.3.4 template<typename E> virtual void decaf::util::StlSet< E >::copy (const StlSet< E > & *source*) [inline, virtual]

Referenced by decaf::util::StlSet< ActiveMQSession * >::StlSet().

6.520.3.5 template<typename E> virtual bool decaf::util::StlSet< E >::equals (const StlSet< E > & *source*) const [inline, virtual]

Compares the passed collection to this one, if they contain the same elements, i.e.

all their elements are equivalent, then it returns true.

Returns:

true if the Collections contain the same elements.

Implements **decaf::util::Collection< E >** (p. 875).

6.520.3.6 template<typename E> virtual bool decaf::util::StlSet< E >::isEmpty () const [inline, virtual]

Returns:

if the set contains any element or not, TRUE or FALSE

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 129).

6.520.3.7 template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () const [inline, virtual]

Implements **decaf::lang::Iterable< E >** (p. 1487).

6.520.3.8 template<typename E> Iterator<E>* decaf::util::StlSet< E >::iterator () [inline, virtual]

Returns:

an iterator over a set of elements of type T.

Implements **decaf::lang::Iterable< E >** (p. 1487).

6.520.3.9 template<typename E> virtual bool decaf::util::StlSet< E >::remove (const E & *value*) throw (lang::exceptions::UnsupportedOperationException, lang::exceptions::IllegalArgumentException) [inline, virtual]

Removes a single instance of the specified element from this collection, if it is present (optional operation).

More formally, removes the first element e such that e == o, if this collection contains one or more such elements. Returns true if this collection contained the specified element (or equivalently, if this collection changed as a result of the call).

This implementation iterates over the collection looking for the specified element. If it finds the element, it removes the element from the collection using the iterator's remove method.

Note that this implementation throws an UnsupportedOperationException if the iterator returned by this collection's iterator method does not implement the remove method and this collection contains the specified object.

Parameters:

value - element to be removed from this collection, if present

Returns:

true if an element was removed as a result of this call

Exceptions:

UnsupportedOperationException if the remove operation is not supported by this collection.

IllegalArgumentException If the value is not a valid entry for this **Collection** (p. 871).

Reimplemented from **decaf::util::AbstractCollection< E >** (p. 130).

6.520.3.10 template<typename E> virtual std::size_t decaf::util::StlSet< E >::size () const [inline, virtual]

Returns:

The number of elements in this set.

Implements **decaf::util::Collection< E >** (p. 878).

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StlSet.h**

6.521 activemq::wireformat::stomp::StompCommandConstants Class Reference

`#include <src/main/activemq/wireformat/stomp/StompCommandConstants.h>`

Static Public Attributes

- static const std::string **CONNECT**
- static const std::string **CONNECTED**
- static const std::string **DISCONNECT**
- static const std::string **SUBSCRIBE**
- static const std::string **UNSUBSCRIBE**
- static const std::string **MESSAGE**
- static const std::string **SEND**
- static const std::string **BEGIN**
- static const std::string **COMMIT**
- static const std::string **ABORT**
- static const std::string **ACK**
- static const std::string **ERROR_CMD**
- static const std::string **RECEIPT**
- static const std::string **HEADER_DESTINATION**
- static const std::string **HEADER_TRANSACTIONID**
- static const std::string **HEADER_CONTENTLENGTH**
- static const std::string **HEADER_SESSIONID**
- static const std::string **HEADER_RECEIPT_REQUIRED**
- static const std::string **HEADER_RECEIPTID**
- static const std::string **HEADER_MESSAGEID**
- static const std::string **HEADER_ACK**
- static const std::string **HEADER_LOGIN**
- static const std::string **HEADER_PASSWORD**
- static const std::string **HEADER_CLIENT_ID**
- static const std::string **HEADER_MESSAGE**
- static const std::string **HEADER_CORRELATIONID**
- static const std::string **HEADER_REQUESTID**
- static const std::string **HEADER_RESPONSEID**
- static const std::string **HEADER_EXPIRES**
- static const std::string **HEADER_PERSISTENT**
- static const std::string **HEADER_REPLYTO**
- static const std::string **HEADER_TYPE**
- static const std::string **HEADER_DISPATCH_ASYNC**
- static const std::string **HEADER_EXCLUSIVE**
- static const std::string **HEADER_MAXPENDINGMSGLIMIT**
- static const std::string **HEADER_NOLOCAL**
- static const std::string **HEADER_PREFETCHSIZE**
- static const std::string **HEADER_JMSPRIORITY**
- static const std::string **HEADER_CONSUMERPRIORITY**
- static const std::string **HEADER_RETROACTIVE**
- static const std::string **HEADER_SUBSCRIPTIONNAME**
- static const std::string **HEADER_OLDSUBSCRIPTIONNAME**

- static const std::string **HEADER_TIMESTAMP**
- static const std::string **HEADER_REDELIVERED**
- static const std::string **HEADER_REDELIVERYCOUNT**
- static const std::string **HEADER_SELECTOR**
- static const std::string **HEADER_ID**
- static const std::string **HEADER_SUBSCRIPTION**
- static const std::string **HEADER_TRANSFORMATION**
- static const std::string **HEADER_TRANSFORMATION_ERROR**
- static const std::string **ACK_CLIENT**
- static const std::string **ACK_AUTO**
- static const std::string **ACK_INDIVIDUAL**
- static const std::string **TEXT**
- static const std::string **BYTES**
- static const std::string **QUEUE_PREFIX**
- static const std::string **TOPIC_PREFIX**
- static const std::string **TEMPQUEUE_PREFIX**
- static const std::string **TEMPTOPIC_PREFIX**

6.521.1 Field Documentation

6.521.1.1 const std::string activemq::wireformat::stomp::StompCommandConstants::ABORT [static]

6.521.1.2 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK [static]

6.521.1.3 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_AUTO [static]

6.521.1.4 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_CLIENT [static]

6.521.1.5 const std::string activemq::wireformat::stomp::StompCommandConstants::ACK_INDIVIDUAL [static]

6.521.1.6 const std::string activemq::wireformat::stomp::StompCommandConstants::BEGIN [static]

6.521.1.7 const std::string activemq::wireformat::stomp::StompCommandConstants::BYTES [static]

6.521.1.8 const std::string activemq::wireformat::stomp::StompCommandConstants::COMMIT [static]

6.521.1.9 const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECT [static]

6.521.1.10 const std::string activemq::wireformat::stomp::StompCommandConstants::CONNECTED [static]

6.521.1.11 const std::string activemq::wireformat::stomp::StompCommandConstants::DISCONNECT [static]

6.521.1.12 const std::string activemq::wireformat::stomp::StompCommandConstants::ERROR_CMD [static]

6.521.1.13 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_ACK [static]

6.521.1.14 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CLIENT_ID [static]

6.521.1.15 const std::string activemq::wireformat::stomp::StompCommandConstants::HEADER_CONSUMERPRIORITY [static]

- src/main/activemq/wireformat/stomp/**StompCommandConstants.h**

6.522 activemq::wireformat::stomp::StompFrame Class Reference

A Stomp-level message frame that encloses all messages to and from the broker.

`#include <src/main/activemq/wireformat/stomp/StompFrame.h>`

Public Member Functions

- **StompFrame** ()
  Default constructor.
- virtual ~**StompFrame** ()
  Destruction.
- **StompFrame** * **clone** () const
  Clonse this message exactly, returns a new instance that the caller is required to delete.
- void **copy** (const **StompFrame** *src)
  Copies the contents of the passed Frame to this one.
- void **setCommand** (const std::string &cmd)
  *Sets the command for this **stomp** (p. 90) frame.*
- const std::string & **getCommand** () const
  Accessor for this frame's command field.
- bool **hasProperty** (const std::string &name) const
  Checks if the given property is present in the Frame.
- std::string **getProperty** (const std::string &name, const std::string &fallback="") const
  Gets a property from this Frame's properties and returns it, or the default value given.
- std::string **removeProperty** (const std::string &name)
  Gets and remove the property specified, if the property is not set, this method returns the empty string.
- void **setProperty** (const std::string &name, const std::string &value)
  Sets the property given to the value specified in this Frame's Properties.
- **decaf::util::Properties** & **getProperties** ()
  Gets access to the header properties for this frame.
- const **decaf::util::Properties** & **getProperties** () const
- const std::vector< unsigned char > & **getBody** () const
  Accessor for the body data of this frame.
- std::vector< unsigned char > & **getBody** ()
  Non-const version of the body accessor.

- std::size_t **getBodyLength** () const

  Return the number of bytes contained in this frames body.

- void **setBody** (const unsigned char *bytes, std::size_t numBytes)

  Sets the body data of this frame as a byte sequence.

- void **toStream** (**decaf::io::DataOutputStream** *stream) const throw (decaf::io::IOException)

  Writes this Frame to an OuputStream in the Stomp Wire Format.

- void **fromStream** (**decaf::io::DataInputStream** *stream) throw (decaf::io::IOException)

  Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

6.522.1 Detailed Description

A Stomp-level message frame that encloses all messages to and from the broker.

6.522.2 Constructor & Destructor Documentation

6.522.2.1 activemq::wireformat::stomp::StompFrame::StompFrame () [inline]

Default constructor.

6.522.2.2 virtual activemq::wireformat::stomp::StompFrame::~StompFrame () [inline, virtual]

Destruction.

6.522.3 Member Function Documentation

6.522.3.1 StompFrame* activemq::wireformat::stomp::StompFrame::clone () const

Clonse this message exactly, returns a new instance that the caller is required to delete.

Returns:

new copy of this message

6.522.3.2 void activemq::wireformat::stomp::StompFrame::copy (const StompFrame * *src*)

Copies the contents of the passed Frame to this one.

Parameters:

src - Frame to copy

6.522.3.3 void activemq::wireformat::stomp::StompFrame::fromStream (decaf::io::DataInputStream * *stream*) throw (decaf::io::IOException)

Reads a Stop Frame from a DataInputStream in the Stomp Wire format.

Parameters:

stream - The stream to read the Frame from.

Exceptions:

IOException if an error occurs while writing the Frame.

6.522.3.4 std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () [inline]

Non-const version of the body accessor.

6.522.3.5 const std::vector<unsigned char>& activemq::wireformat::stomp::StompFrame::getBody () const [inline]

Accessor for the body data of this frame.

Returns:

char pointer to body data

6.522.3.6 std::size_t activemq::wireformat::stomp::StompFrame::getBodyLength () const [inline]

Return the number of bytes contained in this frames body.

Returns:

Body bytes length.

6.522.3.7 const std::string& activemq::wireformat::stomp::StompFrame::getCommand () const [inline]

Accessor for this frame's command field.

6.522.3.8 const decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () const [inline]

6.522.3.9 decaf::util::Properties& activemq::wireformat::stomp::StompFrame::getProperties () [inline]

Gets access to the header properties for this frame.

Returns:

the Properties object owned by this Frame

6.522.3.10 std::string activemq::wireformat::stomp::StompFrame::getProperty (const std::string & *name*, const std::string & *fallback* = "") const [inline]

Gets a property from this Frame's properties and returns it, or the default value given.

Parameters:

name - The name of the property to lookup

fallback - The default value to return if this value isn't set

Returns:

string value of the property asked for.

6.522.3.11 bool activemq::wireformat::stomp::StompFrame::hasProperty (const std::string & *name*) const [inline]

Checks if the given property is present in the Frame.

Parameters:

name - The name of the property to check for.

6.522.3.12 std::string activemq::wireformat::stomp::StompFrame::removeProperty (const std::string & *name*) [inline]

Gets and remove the property specified, if the property is not set, this method returns the empty string.

Parameters:

name - the Name of the property to get and return.

6.522.3.13 void activemq::wireformat::stomp::StompFrame::setBody (const unsigned char * *bytes*, std::size_t *numBytes*)

Sets the body data of this frame as a byte sequence.

Parameters:

bytes The byte buffer to be set in the body.

numBytes The number of bytes in the buffer.

6.522.3.14 void activemq::wireformat::stomp::StompFrame::setCommand (const std::string & *cmd*) [inline]

Sets the command for this **stomp** (p. 90) frame.

Parameters:

cmd command The command to be set.

6.522.3.15 void activemq::wireformat::stomp::StompFrame::setProperty (const std::string & *name*, const std::string & *value*) [inline]

Sets the property given to the value specified in this Frame's Properties.

Parameters:

name - Name of the property.

value - Value to set the property to.

6.522.3.16 void activemq::wireformat::stomp::StompFrame::toStream (decaf::io::DataOutputStream * *stream*) const throw (decaf::io::IOException)

Writes this Frame to an OuputStream in the Stomp Wire Format.

Parameters:

stream - The stream to write the Frame to.

Exceptions:

IOException if an error occurs while reading the Frame.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompFrame.h**

6.523 activemq::wireformat::stomp::StompHelper Class Reference

Utility Methods used when marshaling to and from StompFrame's.

`#include <src/main/activemq/wireformat/stomp/StompHelper.h>`

Public Member Functions

- **StompHelper** ()
- virtual ~**StompHelper** ()
- void **convertProperties** (const **Pointer**< **StompFrame** > &frame, const **Pointer**< **Message** > &message)

  Converts the Headers in a Stomp Frame into Headers in the given Message Command.

- void **convertProperties** (const **Pointer**< **Message** > &message, const **Pointer**< **StompFrame** > &frame)

  Converts the Properties in a Message Command to Valid Headers and Properties in the ***StompFrame*** *(p. 2456).*

- **Pointer**< **ActiveMQDestination** > **convertDestination** (const std::string &destination)

  Converts from a Stomp Destination to an ActiveMQDestination.

- std::string **convertDestination** (const **Pointer**< **ActiveMQDestination** > &destination)

  Converts from a ActiveMQDestination to a Stomp Destination Name.

- std::string **convertMessageId** (const **Pointer**< **MessageId** > &messageId)

  Converts a MessageId instance to a Stomp MessageId String.

- **Pointer**< **MessageId** > **convertMessageId** (const std::string &messageId)

  Converts a Stomp MessageId string to a MessageId.

- std::string **convertConsumerId** (const **Pointer**< **ConsumerId** > &consumerId)

  Converts a ConsumerId instance to a Stomp ConsumerId String.

- **Pointer**< **ConsumerId** > **convertConsumerId** (const std::string &consumerId)

  Converts a Stomp ConsumerId string to a ConsumerId.

- std::string **convertProducerId** (const **Pointer**< **ProducerId** > &producerId)

  Converts a ProducerId instance to a Stomp ProducerId String.

- **Pointer**< **ProducerId** > **convertProducerId** (const std::string &producerId)

  Converts a Stomp ProducerId string to a ProducerId.

- std::string **convertTransactionId** (const **Pointer**< **TransactionId** > &transactionId)

  Converts a TransactionId instance to a Stomp TransactionId String.

- **Pointer**< **TransactionId** > **convertTransactionId** (const std::string &transactionId)

  Converts a Stomp TransactionId string to a TransactionId.

6.523.1 Detailed Description

Utility Methods used when marshaling to and from StompFrame's.

Since:

3.0

6.523.2 Constructor & Destructor Documentation

6.523.2.1 activemq::wireformat::stomp::StompHelper::StompHelper () [inline]

6.523.2.2 virtual activemq::wireformat::stomp::StompHelper::~StompHelper () [inline, virtual]

6.523.3 Member Function Documentation

6.523.3.1 Pointer<ConsumerId> activemq::wireformat::stomp::StompHelper::convertConsumerId (const std::string & *consumerId*)

Converts a Stomp ConsumerId string to a ConsumerId.

Parameters:

consumerId - the String Consumer Id to convert.

Returns:

Pointer to a new ConsumerId.

6.523.3.2 std::string activemq::wireformat::stomp::StompHelper::convertConsumerId (const Pointer< ConsumerId > & *consumerId*)

Converts a ConsumerId instance to a Stomp ConsumerId String.

Parameters:

consumerId - the Consumer instance to convert.

Returns:

a Stomp Consumer Id String.

6.523.3.3 std::string activemq::wireformat::stomp::StompHelper::convertDestination (const Pointer< ActiveMQDestination > & *destination*)

Converts from a ActiveMQDestination to a Stomp Destination Name.

Parameters:

destination - The ActiveMQDestination to Convert

Returns:

the Stomp String name that defines the destination.

6.523.3.4 Pointer<ActiveMQDestination> activemq::wireformat::stomp::StompHelper::convertDestination (const std::string & *destination*)

Converts from a Stomp Destination to an ActiveMQDestination.

Parameters:

destination - The Stomp Destination name string.

Returns:

Pointer to a new ActiveMQDestination.

6.523.3.5 Pointer<MessageId> activemq::wireformat::stomp::StompHelper::convertMessageId (const std::string & *messageId*)

Converts a Stomp MessageId string to a MessageId.

Parameters:

messageId - the String message Id to convert.

Returns:

Pointer to a new MessageId.

6.523.3.6 std::string activemq::wireformat::stomp::StompHelper::convertMessageId (const Pointer< MessageId > & *messageId*)

Converts a MessageId instance to a Stomp MessageId String.

Parameters:

messageId - the MessageId instance to convert.

Returns:

a Stomp Message Id String.

6.523.3.7 Pointer<ProducerId> activemq::wireformat::stomp::StompHelper::convertProducerId (const std::string & *producerId*)

Converts a Stomp ProducerId string to a ProducerId.

Parameters:

producerId - the String Producer Id to convert.

Returns:

Pointer to a new ProducerId.

6.523.3.8 std::string activemq::wireformat::stomp::StompHelper::convertProducerId (const Pointer< ProducerId > & *producerId*)

Converts a ProducerId instance to a Stomp ProducerId String.

Parameters:

producerId - the Producer instance to convert.

Returns:

a Stomp Producer Id String.

6.523.3.9 void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< Message > & *message*, const Pointer< StompFrame > & *frame*)

Converts the Properties in a Message Command to Valid Headers and Properties in the **StompFrame** (p. 2456).

Parameters:

message - The message to move the Headers to.

frame - The frame to extract headers from.

6.523.3.10 void activemq::wireformat::stomp::StompHelper::convertProperties (const Pointer< StompFrame > & *frame*, const Pointer< Message > & *message*)

Converts the Headers in a Stomp Frame into Headers in the given Message Command.

Parameters:

frame - The frame to extract headers from.

message - The message to move the Headers to.

6.523.3.11 Pointer<TransactionId> activemq::wireformat::stomp::StompHelper::convertTransactionId (const std::string & *transactionId*)

Converts a Stomp TransactionId string to a TransactionId.

Parameters:

transactionId - the String Transaction Id to convert.

Returns:

Pointer to a new TransactionId.

6.523.3.12 std::string activemq::wireformat::stomp::StompHelper::convertTransactionId (const Pointer< TransactionId > & *transactionId*)

Converts a TransactionId instance to a Stomp TransactionId String.

Parameters:

transactionId - the Transaction instance to convert.

Returns:

a Stomp Transaction Id String.

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompHelper.h**

6.524 activemq::wireformat::stomp::StompWireFormat Class Reference

`#include <src/main/activemq/wireformat/stomp/StompWireFormat.h>`Inheritance diagram for activemq::wireformat::stomp::StompWireFormat:

Public Member Functions

- **StompWireFormat** ()
- virtual ~**StompWireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out) throw (decaf::io::IOException)

  Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in) throw (decaf::io::IOException)

  Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

- virtual void **setVersion** (int version AMQCPP_UNUSED)

  Set the Version.

- virtual int **getVersion** () const

  Get the Version.

- virtual bool **hasNegotiator** () const

  *Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport) throw (decaf::lang::exceptions::UnsupportedOperationException)

  If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

6.524.1 Constructor & Destructor Documentation

6.524.1.1 activemq::wireformat::stomp::StompWireFormat::StompWireFormat ()

6.524.1.2 virtual activemq::wireformat::stomp::StompWireFormat::~StompWireFormat () [virtual]

6.524.2 Member Function Documentation

6.524.2.1 virtual Pointer<transport::Transport> activemq::wireformat::stomp::StompWireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [virtual]

If the Transport Provides a Negotiator this method will create and return a news instance of the Negotiator.

Returns:

new instance of a **WireFormatNegotiator** (p. 2721).

Exceptions:

UnsupportedOperationException if the **WireFormat** (p. 2693) doesn't have a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.524.2.2 virtual int activemq::wireformat::stomp::StompWireFormat::getVersion () const [inline, virtual]

Get the Version.

Returns:

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.524.2.3 virtual bool activemq::wireformat::stomp::StompWireFormat::hasNegotiator () const [inline, virtual]

Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 2693) provides a Negotiator.

Implements **activemq::wireformat::WireFormat** (p. 2694).

6.524.2.4 virtual void activemq::wireformat::stomp::StompWireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) throw (decaf::io::IOException) [virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal to the output stream.

transport *(*p. ***67****)* The Transport that initiated this marshal call.

out The output stream to write the command to.

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 2695).

6.524.2.5 virtual void activemq::wireformat::stomp::StompWireFormat::setVersion (int version *AMQCPP_UNUSED*) [inline, virtual]

Set the Version.

Parameters:

the version of the wire format

Implements **activemq::wireformat::WireFormat** (p. 2695).

6.524.2.6 virtual Pointer<commands::Command> activemq::wireformat::stomp::StompWireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [virtual]

Stream based un-marshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

Parameters:

transport *(*p. ***67****)* - Pointer to the **transport** (p. 67) that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implements **activemq::wireformat::WireFormat** (p. 2695).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormat.h**

6.525 activemq::wireformat::stomp::StompWireFormatFactory Class Reference

Factory used to create the Stomp Wire Format instance.

`#include <src/main/activemq/wireformat/stomp/StompWireFormatFactory.h>`Inheritance diagram for activemq::wireformat::stomp::StompWireFormatFactory:

Public Member Functions

- **StompWireFormatFactory** ()
- virtual ~**StompWireFormatFactory** ()
- virtual **Pointer< WireFormat >** **createWireFormat** (const **decaf::util::Properties** &properties) throw (decaf::lang::exceptions::IllegalStateException)

  *Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.*

6.525.1 Detailed Description

Factory used to create the Stomp Wire Format instance.

6.525.2 Constructor & Destructor Documentation

6.525.2.1 activemq::wireformat::stomp::StompWireFormatFactory::StompWireFormatFactory () [inline]

6.525.2.2 virtual activemq::wireformat::stomp::StompWireFormatFactory::~StompWireFormatFactory () [inline, virtual]

6.525.3 Member Function Documentation

6.525.3.1 virtual Pointer<WireFormat> activemq::wireformat::stomp::StompWireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalStateException) [virtual]

Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the Properties for this **WireFormat** (p. 2693)

Implements **activemq::wireformat::WireFormatFactory** (p. 2697).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/stomp/**StompWireFormatFactory.h**

6.526 cms::Stoppable Class Reference

Interface for a class that implements the stop method.

`#include <src/main/cms/Stoppable.h>`Inheritance diagram for cms::Stoppable:

Public Member Functions

- virtual ~**Stoppable** ()
- virtual void **stop** ()=0 throw (CMSException)

  Stops this service.

6.526.1 Detailed Description

Interface for a class that implements the stop method. An object that implements this interface implies that it will halt all operations that result in events being propagated to external users, internally the Object can continue to process data but not events will be generated to clients and methods that return data will not return valid results until the object is started again.

Since:

1.0

6.526.2 Constructor & Destructor Documentation

6.526.2.1 virtual cms::Stoppable::~Stoppable () [inline, virtual]

6.526.3 Member Function Documentation

6.526.3.1 virtual void cms::Stoppable::stop () throw (CMSException) [pure virtual]

Stops this service.

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs while stopping the Service.

Implemented in **activemq::core::ActiveMQConnection** (p. 199).

The documentation for this class was generated from the following file:

- src/main/cms/**Stoppable.h**

6.527 decaf::util::logging::StreamHandler Class Reference

`#include <src/main/decaf/util/logging/StreamHandler.h>`Inheritance diagram for decaf::util::logging::StreamHandler:

Public Member Functions

- **StreamHandler** ()

  *Create a **StreamHandler** (p. 2472), with no current output stream.*

- **StreamHandler** (**io::OutputStream** *stream, **Formatter** *formatter)

  *Create a **StreamHandler** (p. 2472), with no current output stream.*

- virtual ~**StreamHandler** ()
- virtual void **close** () throw (cms::CMSException)

  Close the current output stream.

- virtual void **flush** ()

  Flush the Handler's output, clears any buffers.

- virtual void **publish** (const **LogRecord** &record)

  *Publish the Log Record to this **Handler** (p. 1382).*

- virtual void **isLoggable** (const **LogRecord** &record)

  *Check if this **Handler** (p. 1382) would actually log a given **LogRecord** (p. 1645).*

- virtual void **setFilter** (const **Filter** *filter)

  *Sets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.*

- virtual const **Filter** * **getFilter** ()

  *Gets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.*

- virtual void **setLevel** (**Level** level)

  ***Set** (p. 2320) the log level specifying which message levels will be logged by this **Handler** (p. 1382).*

- virtual **Level** **getLevel** ()

  *Get the log level specifying which message levels will be logged by this **Handler** (p. 1382).*

- virtual void **setFormatter** (const **Formatter** *formatter)

  *Sets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).*

- virtual const **Formatter** * **getFormatter** ()

  *Gets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).*

- virtual **io::OutputStream** * **getOutputStream** () const

  *Gets the output Stream that this **Handler** (p. 1382) is using.*

6.527.1 Constructor & Destructor Documentation

6.527.1.1 decaf::util::logging::StreamHandler::StreamHandler () [inline]

Create a **StreamHandler** (p. 2472), with no current output stream.

6.527.1.2 decaf::util::logging::StreamHandler::StreamHandler (io::OutputStream * *stream*, Formatter * *formatter*) [inline]

Create a **StreamHandler** (p. 2472), with no current output stream.

References decaf::util::logging::Fatal.

6.527.1.3 virtual decaf::util::logging::StreamHandler::~StreamHandler () [inline, virtual]

References DECAF_CATCH_NOTHROW, and DECAF_CATCHALL_NOTHROW.

6.527.2 Member Function Documentation

6.527.2.1 virtual void decaf::util::logging::StreamHandler::close () throw (cms::CMSException) [inline, virtual]

Close the current output stream. The close method will perform a flush and then close the **Handler** (p. 1382). After close has been called this **Handler** (p. 1382) should no longer be used. Method calls may either be silently ignored or may throw runtime exceptions.

Exceptions:

CMSException

Implements **decaf::io::Closeable** (p. 840).

6.527.2.2 virtual void decaf::util::logging::StreamHandler::flush () [inline, virtual]

Flush the Handler's output, clears any buffers.

Implements **decaf::util::logging::Handler** (p. 1383).

6.527.2.3 virtual const Filter* decaf::util::logging::StreamHandler::getFilter () [inline, virtual]

Gets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.

Returns:

`Filter` (p. 1314) derived instance

Implements **decaf::util::logging::Handler** (p. 1383).

6.527.2.4 virtual const Formatter* decaf::util::logging::StreamHandler::getFormatter () [inline, virtual]

Gets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).

Returns:

currently configured `Formatter` (p. 1372) derived instance

Implements **decaf::util::logging::Handler** (p. 1383).

6.527.2.5 virtual Level decaf::util::logging::StreamHandler::getLevel () [inline, virtual]

Get the log level specifying which message levels will be logged by this **Handler** (p. 1382).

Returns:

Currently set Level enumeration value

Implements **decaf::util::logging::Handler** (p. 1383).

6.527.2.6 virtual io::OutputStream* decaf::util::logging::StreamHandler::getOutputStream () const [inline, virtual]

Gets the output Stream that this **Handler** (p. 1382) is using.

Returns:

OuputStream pointer used by this handler.

6.527.2.7 virtual void decaf::util::logging::StreamHandler::isLoggable (const LogRecord & *record*) [inline, virtual]

Check if this **Handler** (p. 1382) would actually log a given **LogRecord** (p. 1645).

Parameters:

record `LogRecord` (p. 1645) to check

Implements **decaf::util::logging::Handler** (p. 1383).

6.527.2.8 virtual void decaf::util::logging::StreamHandler::publish (const LogRecord & *record*) [inline, virtual]

Publish the Log Record to this **Handler** (p. 1382).

Parameters:

record The `LogRecord` (p. 1645) to Publish

Implements **decaf::util::logging::Handler** (p. 1384).

References DECAF_CATCH_RETHROW, and DECAF_CATCHALL_THROW.

6.527.2.9 virtual void decaf::util::logging::StreamHandler::setFilter (const Filter * *filter*) [inline, virtual]

Sets the **Filter** (p. 1314) that this **Handler** (p. 1382) uses to filter Log Records.

Parameters:

filter `Filter` (p. 1314) derived instance

Implements **decaf::util::logging::Handler** (p. 1384).

6.527.2.10 virtual void decaf::util::logging::StreamHandler::setFormatter (const Formatter * *formatter*) [inline, virtual]

Sets the `Formatter` (p. 1372) used by this **Handler** (p. 1382).

Parameters:

formatter `Formatter` (p. 1372) derived instance

Implements **decaf::util::logging::Handler** (p. 1384).

6.527.2.11 virtual void decaf::util::logging::StreamHandler::setLevel (Level *level*) [inline, virtual]

Set (p. 2320) the log level specifying which message levels will be logged by this **Handler** (p. 1382). The intention is to allow developers to turn on verbose **logging** (p. 120), but to limit the messages that are sent to certain Handlers.

Parameters:

level Level enumeration value

Implements **decaf::util::logging::Handler** (p. 1384).

The documentation for this class was generated from the following file:

- src/main/decaf/util/logging/**StreamHandler.h**

6.528 cms::StreamMessage Class Reference

Interface for a **StreamMessage** (p. 2476).

`#include <src/main/cms/StreamMessage.h>`Inheritance diagram for cms::StreamMessage:

Public Member Functions

- virtual ~**StreamMessage** ()
- virtual bool **readBoolean** () const =0 throw (cms::CMSException)

  Reads a Boolean from the Stream message stream.

- virtual void **writeBoolean** (bool value)=0 throw (cms::CMSException)

  Writes a boolean to the Stream message stream as a 1-byte value.

- virtual unsigned char **readByte** () const =0 throw (cms::CMSException)

  Reads a Byte from the Stream message stream.

- virtual void **writeByte** (unsigned char value)=0 throw (cms::CMSException)

  Writes a byte to the Stream message stream as a 1-byte value.

- virtual std::size_t **readBytes** (std::vector< unsigned char > &value) const =0 throw (cms::CMSException)

  Reads a byte array from the Stream message stream.

- virtual void **writeBytes** (const std::vector< unsigned char > &value)=0 throw (cms::CMSException)

  Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

- virtual std::size_t **readBytes** (unsigned char *&buffer, std::size_t length) const =0 throw (cms::CMSException)

  Reads a portion of the Stream message stream.

- virtual void **writeBytes** (const unsigned char *value, std::size_t offset, std::size_t length)=0 throw (cms::CMSException)

  Writes a portion of a byte array to the Stream message stream.

- virtual char **readChar** () const =0 throw (cms::CMSException)

  Reads a Char from the Stream message stream.

- virtual void **writeChar** (char value)=0 throw (cms::CMSException)

  Writes a char to the Stream message stream as a 1-byte value.

- virtual float **readFloat** () const =0 throw (cms::CMSException)

  Reads a 32 bit float from the Stream message stream.

- virtual void **writeFloat** (float value)=0 throw (cms::CMSException)

Writes a float to the Stream message stream as a 4 byte value.

- virtual double **readDouble** () const =0 throw (cms::CMSException)
  Reads a 64 bit double from the Stream message stream.
- virtual void **writeDouble** (double value)=0 throw (cms::CMSException)
  Writes a double to the Stream message stream as a 8 byte value.
- virtual short **readShort** () const =0 throw (cms::CMSException)
  Reads a 16 bit signed short from the Stream message stream.
- virtual void **writeShort** (short value)=0 throw (cms::CMSException)
  Writes a signed short to the Stream message stream as a 2 byte value.
- virtual unsigned short **readUnsignedShort** () const =0 throw (cms::CMSException)
  Reads a 16 bit unsigned short from the Stream message stream.
- virtual void **writeUnsignedShort** (unsigned short value)=0 throw (cms::CMSException)
  Writes a unsigned short to the Stream message stream as a 2 byte value.
- virtual int **readInt** () const =0 throw (cms::CMSException)
  Reads a 32 bit signed integer from the Stream message stream.
- virtual void **writeInt** (int value)=0 throw (cms::CMSException)
  Writes a signed int to the Stream message stream as a 4 byte value.
- virtual long long **readLong** () const =0 throw (cms::CMSException)
  Reads a 64 bit long from the Stream message stream.
- virtual void **writeLong** (long long value)=0 throw (cms::CMSException)
  Writes a long long to the Stream message stream as a 8 byte value.
- virtual std::string **readString** () const =0 throw (cms::CMSException)
  Reads an ASCII String from the Stream message stream.
- virtual void **writeString** (const std::string &value)=0 throw (cms::CMSException)
  Writes an ASCII String to the Stream message stream.

6.528.1 Detailed Description

Interface for a **StreamMessage** (p. 2476). The stream Messages provides a **Message** (p. 1753) type whose body is a stream of self describing primitive types. The primitive values are read and written using accessors specific to the given types.

StreamMessage (p. 2476) objects support the following conversion table. The marked cases must be supported. The unmarked cases must throw a **CMSException** (p. 850). The string-to-primitive conversions may throw a runtime exception if the primitive's valueOf() method does not accept it as a valid String representation of the primitive.

A value written as the row type can be read as the column type.

	boolean	byte	short	char	int	long	float	double	String	byte[]
boolean	X								X	
byte		X	X		X	X			X	
short			X		X	X			X	
char				X					X	
int					X	X			X	
long						X			X	
float							X	X	X	
double								X	X	
String	X	X	X		X	X	X	X	X	
byte[]										X

Since:

1.3

6.528.2 Constructor & Destructor Documentation

6.528.2.1 virtual cms::StreamMessage::~StreamMessage () [inline, virtual]

6.528.3 Member Function Documentation

6.528.3.1 virtual bool cms::StreamMessage::readBoolean () const throw (cms::CMSException) [pure virtual]

Reads a Boolean from the Stream message stream.

Returns:

boolean value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 378).

6.528.3.2 virtual unsigned char cms::StreamMessage::readByte () const throw (cms::CMSException) [pure virtual]

Reads a Byte from the Stream message stream.

Returns:

unsigned char value from stream

Exceptions:

CMSException *(*p. ***850)*** - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841)*** - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842)*** - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876)*** - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 378).

6.528.3.3 virtual std::size_t cms::StreamMessage::readBytes (unsigned char *&* *buffer*, std::size_t *length*) const throw (cms::CMSException) [pure virtual]

Reads a portion of the Stream message stream. If the length of array value is less than the number of bytes remaining to be read from the stream, the array should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of array value, the bytes should be read into the array. The return value of the total number of bytes read will be less than the length of the array, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

If length is negative, or length is greater than the length of the array value, then an **CMSException** (p. 850) is thrown. No bytes will be read from the stream for this exception case.

Parameters:

buffer the buffer into which the data is read

length the number of bytes to read; must be less than or equal to value.length

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException *(*p. ***850)*** - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841)*** - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842)*** - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876)*** - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 379).

6.528.3.4 virtual std::size_t cms::StreamMessage::readBytes (std::vector< unsigned char > & *value*) const throw (cms::CMSException) [pure virtual]

Reads a byte array from the Stream message stream. If the length of vector value is less than the number of bytes remaining to be read from the stream, the vector should be filled. A subsequent call reads the next increment, and so on.

If the number of bytes remaining in the stream is less than the length of vector value, the bytes should be read into the vector. The return value of the total number of bytes read will be less than the length of the vector, indicating that there are no more bytes left to be read from the stream. The next read of the stream returns -1.

Parameters:

value buffer to place data in

Returns:

the total number of bytes read into the buffer, or -1 if there is no more data because the end of the stream has been reached

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 379).

6.528.3.5 virtual char cms::StreamMessage::readChar () const throw (cms::CMSException) [pure virtual]

Reads a Char from the Stream message stream.

Returns:

char value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 380).

6.528.3.6 virtual double cms::StreamMessage::readDouble () const throw (cms::CMSException) [pure virtual]

Reads a 64 bit double from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 380).

6.528.3.7 virtual float cms::StreamMessage::readFloat () const throw (cms::CMSException) [pure virtual]

Reads a 32 bit float from the Stream message stream.

Returns:

double value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 380).

6.528.3.8 virtual int cms::StreamMessage::readInt () const throw (cms::CMSException) [pure virtual]

Reads a 32 bit signed integer from the Stream message stream.

Returns:

int value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 381).

6.528.3.9 virtual long long cms::StreamMessage::readLong () const throw (cms::CMSException) [pure virtual]

Reads a 64 bit long from the Stream message stream.

Returns:

long long value from stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of message stream has been reached.

MessageFormatException (p. ***1842***) - if this type conversion is invalid.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 381).

6.528.3.10 virtual short cms::StreamMessage::readShort () const throw (cms::CMSException) [pure virtual]

Reads a 16 bit signed short from the Stream message stream.

Returns:

short value from stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException (p. ***1841***) - if unexpected end of message stream has been reached.

MessageFormatException (p. ***1842***) - if this type conversion is invalid.

MessageNotReadableException (p. ***1876***) - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 381).

6.528.3.11 virtual std::string cms::StreamMessage::readString () const throw (cms::CMSException) [pure virtual]

Reads an ASCII String from the Stream message stream.

Returns:

String from stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 381).

6.528.3.12 virtual unsigned short cms::StreamMessage::readUnsignedShort () const throw (cms::CMSException) [pure virtual]

Reads a 16 bit unsigned short from the Stream message stream.

Returns:

unsigned short value from stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to read the message due to some internal error.

MessageEOFException *(*p. ***1841****)* - if unexpected end of message stream has been reached.

MessageFormatException *(*p. ***1842****)* - if this type conversion is invalid.

MessageNotReadableException *(*p. ***1876****)* - if the message is in write-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 382).

6.528.3.13 virtual void cms::StreamMessage::writeBoolean (bool *value*) throw (cms::CMSException) [pure virtual]

Writes a boolean to the Stream message stream as a 1-byte value. The value true is written as the value (byte)1; the value false is written as the value (byte)0.

Parameters:

value boolean to write to the stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(*p. ***1877****)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 382).

6.528.3.14 virtual void cms::StreamMessage::writeByte (unsigned char *value*) throw (cms::CMSException) [pure virtual]

Writes a byte to the Stream message stream as a 1-byte value.

Parameters:

value byte to write to the stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(*p. ***1877****)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 383).

6.528.3.15 virtual void cms::StreamMessage::writeBytes (const unsigned char * *value*, std::size_t *offset*, std::size_t *length*) throw (cms::CMSException) [pure virtual]

Writes a portion of a byte array to the Stream message stream. size as the number of bytes to write.

Parameters:

value bytes to write to the stream

offset the initial offset within the byte array

length the number of bytes to use

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(*p. ***1877****)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 383).

6.528.3.16 virtual void cms::StreamMessage::writeBytes (const std::vector< unsigned char > & *value*) throw (cms::CMSException) [pure virtual]

Writes a byte array to the Stream message stream using the vector size as the number of bytes to write.

Parameters:

value bytes to write to the stream

Exceptions:

CMSException *(*p. ***850****)* - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException *(*p. ***1877****)* - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 383).

6.528.3.17 virtual void cms::StreamMessage::writeChar (char *value*) throw (cms::CMSException) [pure virtual]

Writes a char to the Stream message stream as a 1-byte value.

Parameters:

value char to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 384).

6.528.3.18 virtual void cms::StreamMessage::writeDouble (double *value*) throw (cms::CMSException) [pure virtual]

Writes a double to the Stream message stream as a 8 byte value.

Parameters:

value double to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 384).

6.528.3.19 virtual void cms::StreamMessage::writeFloat (float *value*) throw (cms::CMSException) [pure virtual]

Writes a float to the Stream message stream as a 4 byte value.

Parameters:

value float to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 384).

6.528.3.20 virtual void cms::StreamMessage::writeInt (int *value*) throw (cms::CMSException) [pure virtual]

Writes a signed int to the Stream message stream as a 4 byte value.

Parameters:

value signed int to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 384).

6.528.3.21 virtual void cms::StreamMessage::writeLong (long long *value*) throw (cms::CMSException) [pure virtual]

Writes a long long to the Stream message stream as a 8 byte value.

Parameters:

value signed long long to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 385).

6.528.3.22 virtual void cms::StreamMessage::writeShort (short *value*) throw (cms::CMSException) [pure virtual]

Writes a signed short to the Stream message stream as a 2 byte value.

Parameters:

value signed short to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 385).

6.528.3.23 virtual void cms::StreamMessage::writeString (const std::string & *value*) throw (cms::CMSException) [pure virtual]

Writes an ASCII String to the Stream message stream.

Parameters:

value String to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 385).

6.528.3.24 virtual void cms::StreamMessage::writeUnsignedShort (unsigned short *value*) throw (cms::CMSException) [pure virtual]

Writes a unsigned short to the Stream message stream as a 2 byte value.

Parameters:

value unsigned short to write to the stream

Exceptions:

CMSException (p. ***850***) - if the CMS provider fails to write the message due to some internal error.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode.

Implemented in **activemq::commands::ActiveMQStreamMessage** (p. 385).

The documentation for this class was generated from the following file:

- src/main/cms/**StreamMessage.h**

6.529 decaf::util::StringTokenizer Class Reference

`#include <src/main/decaf/util/StringTokenizer.h>`

Public Member Functions

- **StringTokenizer** (const std::string &str, const std::string &delim=" \t\n\r\f", bool returnDelims=false)

  Constructs a string tokenizer for the specified string.

- virtual ~**StringTokenizer** ()
- virtual int **countTokens** () const

  Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception.

- virtual bool **hasMoreTokens** () const

  Tests if there are more tokens available from this tokenizer's string.

- virtual std::string **nextToken** () throw (lang::exceptions::NoSuchElementException)

  Returns the next token from this string tokenizer.

- virtual std::string **nextToken** (const std::string &delim) throw (lang::exceptions::NoSuchElementException)

  Returns the next token in this string tokenizer's string.

- virtual unsigned int **toArray** (std::vector< std::string > &array)

  Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

- virtual void **reset** (const std::string &str="", const std::string &delim="", bool returnDelims=false)

  Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning.

6.529.1 Constructor & Destructor Documentation

6.529.1.1 decaf::util::StringTokenizer::StringTokenizer (const std::string & *str*, const std::string & *delim* = " \t\n\r\f", bool *returnDelims* = false)

Constructs a string tokenizer for the specified string. All characters in the delim argument are the delimiters for separating tokens.

If the returnDelims flag is true, then the delimiter characters are also returned as tokens. Each delimiter is returned as a string of length one. If the flag is false, the delimiter characters are skipped and only serve as separators between tokens.

Note that if delim is "", this constructor does not throw an exception. However, trying to invoke other methods on the resulting **StringTokenizer** (p. 2488) may result in an Exception.

Parameters:

str - The string to tokenize

delim - String containing the delimiters

returnDelims - boolean indicating if the delimiters are returned as tokens

6.529.1.2 virtual decaf::util::StringTokenizer::~StringTokenizer () [virtual]

6.529.2 Member Function Documentation

6.529.2.1 virtual int decaf::util::StringTokenizer::countTokens () const [virtual]

Calculates the number of times that this tokenizer's nextToken method can be called before it generates an exception. The current position is not advanced.

Returns:

Count of remaining tokens

6.529.2.2 virtual bool decaf::util::StringTokenizer::hasMoreTokens () const [virtual]

Tests if there are more tokens available from this tokenizer's string.

Returns:

true if there are more tokens remaining

6.529.2.3 virtual std::string decaf::util::StringTokenizer::nextToken (const std::string & *delim*) throw (lang::exceptions::NoSuchElementException) [virtual]

Returns the next token in this string tokenizer's string. First, the set of characters considered to be delimiters by this **StringTokenizer** (p. 2488) object is changed to be the characters in the string delim. Then the next token in the string after the current position is returned. The current position is advanced beyond the recognized token. The new delimiter set remains the default after this call.

Parameters:

delim - string containing the new set of delimiters

Returns:

next string in the token list

Exceptions:

NoSuchElementException

6.529.2.4 virtual std::string decaf::util::StringTokenizer::nextToken () throw (lang::exceptions::NoSuchElementException) [virtual]

Returns the next token from this string tokenizer.

Returns:

string value of next token

Exceptions:

NoSuchElementException

6.529.2.5 virtual void decaf::util::StringTokenizer::reset (const std::string & *str* = "", const std::string & *delim* = "", bool *returnDelims* = false) [virtual]

Resets the Tokenizer's position in the String to the Beginning calls to countToken and nextToken now start back at the beginning. This allows this object to be reused, the caller need not create a new instance every time a String needs tokenizing. If set the string param will reset the string that this Tokenizer is working on. If set to "" no change is made. If set the delim param will reset the string that this Tokenizer is using to tokenizer the string. If set to "", no change is made If set the return Delims will set if this Tokenizer will return delimiters as tokens. Defaults to false.

Parameters:

str - New String to tokenize or "", defaults to ""

delim - New Delimiter String to use or "", defaults to ""

returnDelims - Should the Tokenizer return delimiters as Tokens, default false

6.529.2.6 virtual unsigned int decaf::util::StringTokenizer::toArray (std::vector< std::string > & *array*) [virtual]

Grab all remaining tokens in the String and return them in the vector that is passed in by reference.

Parameters:

array - vector to place token strings in

Returns:

number of string placed into the vector

The documentation for this class was generated from the following file:

- src/main/decaf/util/**StringTokenizer.h**

6.530 activemq::commands::SubscriptionInfo Class Reference

`#include <src/main/activemq/commands/SubscriptionInfo.h>`Inheritance diagram for activemq::commands::SubscriptionInfo:

Public Member Functions

- **SubscriptionInfo** ()
- virtual ~**SubscriptionInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **SubscriptionInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual const std::string & **getClientId** () const
- virtual std::string & **getClientId** ()
- virtual void **setClientId** (const std::string &**clientId**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getDestination** ()
- virtual void **setDestination** (const **Pointer**< **ActiveMQDestination** > &**destination**)
- virtual const std::string & **getSelector** () const
- virtual std::string & **getSelector** ()
- virtual void **setSelector** (const std::string &**selector**)
- virtual const std::string & **getSubcriptionName** () const
- virtual std::string & **getSubcriptionName** ()
- virtual void **setSubcriptionName** (const std::string &**subcriptionName**)
- virtual const **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** () const
- virtual **Pointer**< **ActiveMQDestination** > & **getSubscribedDestination** ()
- virtual void **setSubscribedDestination** (const **Pointer**< **ActiveMQDestination** > &**subscribedDestination**)

Static Public Attributes

- static const unsigned char **ID_SUBSCRIPTIONINFO** = 55

Protected Member Functions

- **SubscriptionInfo** (const **SubscriptionInfo** &)
- **SubscriptionInfo** & **operator**= (const **SubscriptionInfo** &)

Protected Attributes

- std::string **clientId**
- **Pointer**< **ActiveMQDestination** > **destination**
- std::string **selector**
- std::string **subcriptionName**
- **Pointer**< **ActiveMQDestination** > **subscribedDestination**

6.530.1 Constructor & Destructor Documentation

6.530.1.1 activemq::commands::SubscriptionInfo::SubscriptionInfo (const SubscriptionInfo &) [inline, protected]

6.530.1.2 activemq::commands::SubscriptionInfo::SubscriptionInfo ()

6.530.1.3 virtual activemq::commands::SubscriptionInfo::~SubscriptionInfo () [virtual]

6.530.2 Member Function Documentation

6.530.2.1 virtual SubscriptionInfo* activemq::commands::SubscriptionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.530.2.2 virtual void activemq::commands::SubscriptionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

6.530.2.3 virtual bool activemq::commands::SubscriptionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

6.530.2.4 virtual std::string& activemq::commands::SubscriptionInfo::getClientId () [virtual]

6.530.2.5 virtual const std::string& activemq::commands::SubscriptionInfo::getClientId () const [virtual]

6.530.2.6 virtual unsigned char activemq::commands::SubscriptionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.530.2.7 virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () [virtual]

6.530.2.8 virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getDestination () const [virtual]

6.530.2.9 virtual std::string& activemq::commands::SubscriptionInfo::getSelector () [virtual]

6.530.2.10 virtual const std::string& activemq::commands::SubscriptionInfo::getSelector () const [virtual]

6.530.2.11 virtual std::string& activemq::commands::SubscriptionInfo::getSubcriptionName () [virtual]

6.530.2.12 virtual const std::string& activemq::commands::SubscriptionInfo::getSubcriptionName () const [virtual]

6.530.2.13 virtual Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () [virtual]

6.530.2.14 virtual const Pointer<ActiveMQDestination>& activemq::commands::SubscriptionInfo::getSubscribedDestination () const [virtual]

6.530.2.15 SubscriptionInfo& activemq::commands::SubscriptionInfo::operator= (const SubscriptionInfo &) [inline, protected]

6.530.2.16 virtual void activemq::commands::SubscriptionInfo::setClientId (const std::string & *clientId*) [virtual]

6.530.2.17 virtual void activemq::commands::SubscriptionInfo::setDestination (const Pointer< ActiveMQDestination > & *destination*) [virtual]

6.530.2.18 virtual void activemq::commands::SubscriptionInfo::setSelector (const std::string & *selector*) [virtual]

6.530.2.19 virtual void activemq::commands::SubscriptionInfo::setSubcriptionName (const std::string & *subcriptionName*) [virtual]

6.530.2.20 virtual void activemq::commands::SubscriptionInfo::setSubscribedDestination (const Pointer< ActiveMQDestination > & *subscribedDestination*) [virtual]

6.530.2.21 virtual std::string activemq::commands::SubscriptionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

6.530.3 Field Documentation

6.530.3.1 std::string activemq::commands::SubscriptionInfo::clientId [protected]

6.530.3.2 Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::destination [protected]

6.530.3.3 const unsigned char activemq::commands::SubscriptionInfo::ID_SUBSCRIPTIONINFO = 55 [static]

6.530.3.4 std::string activemq::commands::SubscriptionInfo::selector [protected]

6.530.3.5 std::string activemq::commands::SubscriptionInfo::subcriptionName [protected]

6.530.3.6 Pointer<ActiveMQDestination> activemq::commands::SubscriptionInfo::subscribedDestination [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**SubscriptionInfo.h**

6.531 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMar Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2496).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/SubscriptionInfoMarshaller.h>`Inheritan diagram for activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual ~**SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.531.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2496). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.531.2 Constructor & Destructor Documentation

6.531.2.1 activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::SubscriptionInfo () [inline]

6.531.2.2 virtual activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::~SubscriptionIn () [inline, virtual]

6.531.3 Member Function Documentation

6.531.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.531.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.531.3.3 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.531.3.4 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.531.3.5 virtual int activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.531.3.6 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.531.3.7 virtual void activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**SubscriptionInfoMarshaller.h**

6.532 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMar Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2500).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/SubscriptionInfoMarshaller.h>`Inheritan diagram for activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual ~**SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.532.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2500). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.532.2 Constructor & Destructor Documentation

6.532.2.1 activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::SubscriptionInfo () [inline]

6.532.2.2 virtual activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::~SubscriptionIn () [inline, virtual]

6.532.3 Member Function Documentation

6.532.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.532.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.532.3.3 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.532.3.4 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.532.3.5 virtual int activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.532.3.6 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.532.3.7 virtual void activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**SubscriptionInfoMarshaller.h**

6.533 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMar Class Reference

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2504).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/SubscriptionInfoMarshaller.h>`Inheritan diagram for activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller:

Public Member Functions

- **SubscriptionInfoMarshaller** ()
- virtual ~**SubscriptionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.533.1 Detailed Description

Marshaling code for Open Wire Format for **SubscriptionInfoMarshaller** (p. 2504). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.533.2 Constructor & Destructor Documentation

6.533.2.1 activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::SubscriptionInfo () [inline]

6.533.2.2 virtual activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::~SubscriptionIn () [inline, virtual]

6.533.3 Member Function Documentation

6.533.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.533.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.533.3.3 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.533.3.4 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.533.3.5 virtual int activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.533.3.6 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.533.3.7 virtual void activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**SubscriptionInfoMarshaller.h**

6.534 decaf::util::concurrent::Synchronizable Class Reference

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

`#include <src/main/decaf/util/concurrent/Synchronizable.h>`Inheritance diagram for decaf::util::concurrent::Synchronizable:

Public Member Functions

- virtual ~**Synchronizable** ()
- virtual void **lock** ()=0 throw (lang::Exception)
  Locks the object.
- virtual void **unlock** ()=0 throw (lang::Exception)
  Unlocks the object.
- virtual void **wait** ()=0 throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **wait** (unsigned long millisecs)=0 throw (lang::Exception)
  Waits on a signal from this object, which is generated by a call to Notify.
- virtual void **notify** ()=0 throw (lang::Exception)
  Signals a waiter on this object that it can now wake up and continue.
- virtual void **notifyAll** ()=0 throw (lang::Exception)
  Signals the waiters on this object that it can now wake up and continue.

6.534.1 Detailed Description

The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

6.534.2 Constructor & Destructor Documentation

6.534.2.1 virtual decaf::util::concurrent::Synchronizable::~Synchronizable () `[inline, virtual]`

6.534.3 Member Function Documentation

6.534.3.1 virtual void decaf::util::concurrent::Synchronizable::lock () throw (lang::Exception) `[pure virtual]`

Locks the object.

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1808), **decaf::internal::io::StandardErrorOutputStream** (p. 2400), **decaf::internal::io::StandardInputStream** (p. 2404), **decaf::internal::io::StandardOutputStream** (p. 2410), **decaf::io::BlockingByteArrayInputStream** (p. 568), **decaf::io::ByteArrayInputStream** (p. 718), **decaf::io::ByteArrayOutputStream** (p. 725), **decaf::io::FilterInputStream** (p. 1317), **decaf::io::FilterOutputStream** (p. 1324), **decaf::net::SocketInputStream** (p. 2386), **decaf::net::SocketOutputStream** (p. 2391), **decaf::util::AbstractCollection< E >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 926), **decaf::util::concurrent::Mutex** (p. 1922), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2436), **decaf::util::StlQueue< T >** (p. 2443), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 129), **decaf::util::AbstractCollection< CompositeTask * >** (p. 129), **decaf::util::AbstractCollection< URI >** (p. 129), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 129), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 129), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 129), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 129), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 926), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2436), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2436), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2436), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2436), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2436), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2436), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2443), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2443), **decaf::util::StlQueue< Task >** (p. 2443), **decaf::util::StlQueue< Pointer< Command > >** (p. 2443), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2443).

Referenced by decaf::util::concurrent::Lock::lock().

6.534.3.2 virtual void decaf::util::concurrent::Synchronizable::notify () throw (lang::Exception) [pure virtual]

Signals a waiter on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1808), **decaf::internal::io::StandardErrorOutputStream** (p. 2400), **decaf::internal::io::StandardInputStream** (p. 2405), **decaf::internal::io::StandardOutputStream** (p. 2410), **decaf::io::BlockingByteArrayInputStream** (p. 569), **decaf::io::ByteArrayInputStream** (p. 719), **decaf::io::ByteArrayOutputStream** (p. 725), **decaf::io::FilterInputStream** (p. 1318), **decaf::io::FilterOutputStream** (p. 1325), **decaf::net::SocketInputStream** (p. 2386), **decaf::net::SocketOutputStream** (p. 2392), **decaf::util::AbstractCollection< E >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 926), **decaf::util::concurrent::Mutex** (p. 1922), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2436), **decaf::util::StlQueue< T >** (p. 2443), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 129), **decaf::util::AbstractCollection< CompositeTask * >** (p. 129), **decaf::util::AbstractCollection< URI >** (p. 129), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 129), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 129), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 129), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 129), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 926), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2436), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2436), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2436), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2436), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2436), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2436), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2443), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2443), **decaf::util::StlQueue< Task >** (p. 2443), **decaf::util::StlQueue<**

Pointer< Command > > (p. 2443), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2443).

6.534.3.3 virtual void decaf::util::concurrent::Synchronizable::notifyAll () throw (lang::Exception) [pure virtual]

Signals the waiters on this object that it can now wake up and continue. Must have this object locked before calling.

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1808), **decaf::internal::io::StandardErrorOutputStream** (p. 2401), **decaf::internal::io::StandardInputStream** (p. 2405), **decaf::internal::io::StandardOutputStream** (p. 2410), **decaf::io::BlockingByteArrayInputStream** (p. 569), **decaf::io::ByteArrayInputStream** (p. 719), **decaf::io::ByteArrayOutputStream** (p. 725), **decaf::io::FilterInputStream** (p. 1318), **decaf::io::FilterOutputStream** (p. 1325), **decaf::net::SocketInputStream** (p. 2387), **decaf::net::SocketOutputStream** (p. 2392), **decaf::util::AbstractCollection< E >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 926), **decaf::util::concurrent::Mutex** (p. 1922), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2436), **decaf::util::StlQueue< T >** (p. 2444), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 129), **decaf::util::AbstractCollection< CompositeTask * >** (p. 129), **decaf::util::AbstractCollection< URI >** (p. 129), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 129), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 129), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 129), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 129), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 129), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 926), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 926), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2436), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2436), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2436), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2436), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2436), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2436), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2436),

decaf::util::StlMap< std::string, cms::Topic * > (p. 2436), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2444), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2444), **decaf::util::StlQueue< Task >** (p. 2444), **decaf::util::StlQueue< Pointer< Command > >** (p. 2444), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2444).

6.534.3.4 virtual void decaf::util::concurrent::Synchronizable::unlock () throw (lang::Exception) [pure virtual]

Unlocks the object.

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1809), **decaf::internal::io::StandardInputStream** (p. 2407), **decaf::internal::io::StandardOutputStream** (p. 2411), **decaf::io::BlockingByteArrayInputStream** (p. 571), **decaf::io::ByteArrayInputStream** (p. 721), **decaf::io::ByteArrayOutputStream** (p. 726), **decaf::io::FilterInputStream** (p. 1321), **decaf::io::FilterOutputStream** (p. 1325), **decaf::net::SocketInputStream** (p. 2388), **decaf::net::SocketOutputStream** (p. 2392), **decaf::util::AbstractCollection< E >** (p. 132), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 930), **decaf::util::concurrent::Mutex** (p. 1923), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2438), **decaf::util::StlQueue< T >** (p. 2445), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 132), **decaf::util::AbstractCollection< CompositeTask * >** (p. 132), **decaf::util::AbstractCollection< URI >** (p. 132), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 132), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 132), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 132), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 132), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 132), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 930), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 930), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2438), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2438), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2438), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2438), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2438), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2438), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2438), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR**

> (p. 2438), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2438), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2438), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2445), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2445), **decaf::util::StlQueue< Task >** (p. 2445), **decaf::util::StlQueue< Pointer< Command > >** (p. 2445), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2445).

Referenced by decaf::util::concurrent::Lock::unlock(), and decaf::util::concurrent::Lock::~Lock().

6.534.3.5 virtual void decaf::util::concurrent::Synchronizable::wait (unsigned long *millisecs*) throw (lang::Exception) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling. This wait will timeout after the specified time interval.

Parameters:

millisecs the time in millisecsonds to wait, or WAIT_INIFINITE

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1809), **decaf::internal::io::StandardErrorOutputStream** (p. 2401), **decaf::internal::io::StandardInputStream** (p. 2407), **decaf::internal::io::StandardOutputStream** (p. 2411), **decaf::io::BlockingByteArrayInputStream** (p. 571), **decaf::io::ByteArrayInputStream** (p. 721), **decaf::io::ByteArrayOutputStream** (p. 727), **decaf::io::FilterInputStream** (p. 1321), **decaf::io::FilterOutputStream** (p. 1325), **decaf::net::SocketInputStream** (p. 2388), **decaf::net::SocketOutputStream** (p. 2392), **decaf::util::AbstractCollection< E >** (p. 132), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 931), **decaf::util::concurrent::Mutex** (p. 1923), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2439), **decaf::util::StlQueue< T >** (p. 2445), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 132), **decaf::util::AbstractCollection< CompositeTask * >** (p. 132), **decaf::util::AbstractCollection< URI >** (p. 132), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 132), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 132), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 132), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 132), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 132), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR >** (p. 931), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2439), **decaf::util::StlMap< std::string,**

WireFormatFactory * > (p. 2439), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2439), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2439), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2439), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2439), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2445), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2445), **decaf::util::StlQueue< Task >** (p. 2445), **decaf::util::StlQueue< Pointer< Command > >** (p. 2445), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2445).

6.534.3.6 virtual void decaf::util::concurrent::Synchronizable::wait () throw (lang::Exception) [pure virtual]

Waits on a signal from this object, which is generated by a call to Notify. Must have this object locked before calling.

Exceptions:

Exception

Implemented in **activemq::core::MessageDispatchChannel** (p. 1809), **decaf::internal::io::StandardErrorOutputStream** (p. 2401), **decaf::internal::io::StandardInputStream** (p. 2408), **decaf::internal::io::StandardOutputStream** (p. 2411), **decaf::io::BlockingByteArrayInputStream** (p. 572), **decaf::io::ByteArrayInputStream** (p. 722), **decaf::io::ByteArrayOutputStream** (p. 727), **decaf::io::FilterInputStream** (p. 1321), **decaf::io::FilterOutputStream** (p. 1326), **decaf::net::SocketInputStream** (p. 2389), **decaf::net::SocketOutputStream** (p. 2393), **decaf::util::AbstractCollection< E >** (p. 133), **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >** (p. 931), **decaf::util::concurrent::Mutex** (p. 1923), **decaf::util::StlMap< K, V, COMPARATOR >** (p. 2439), **decaf::util::StlQueue< T >** (p. 2445), **decaf::util::AbstractCollection< Pointer< Synchronization > >** (p. 133), **decaf::util::AbstractCollection< CompositeTask * >** (p. 133), **decaf::util::AbstractCollection< URI >** (p. 133), **decaf::util::AbstractCollection< ActiveMQSession * >** (p. 133), **decaf::util::AbstractCollection< Pointer< DestinationInfo > >** (p. 133), **decaf::util::AbstractCollection< PrimitiveValueNode >** (p. 133), **decaf::util::AbstractCollection< Pointer< Command > >** (p. 133), **decaf::util::AbstractCollection< Pointer< BackupTransport > >** (p. 133), **decaf::util::concurrent::ConcurrentStlMap< Pointer< TransactionId >, Pointer< TransactionState >, TransactionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< MessageId >, Pointer< Message >, MessageId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConnectionId >, Pointer< ConnectionState >, ConnectionId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< ConsumerId >, Pointer< ConsumerState >, ConsumerId::COMPARATOR >** (p. 931), **decaf::util::concurrent::ConcurrentStlMap< Pointer< SessionId >, Pointer< SessionState >, SessionId::COMPARATOR >** (p. 931),

decaf::util::concurrent::ConcurrentStlMap< Pointer< ProducerId >, Pointer< ProducerState >, ProducerId::COMPARATOR > (p. 931), **decaf::util::StlMap< cms::Session *, SessionResolver * >** (p. 2439), **decaf::util::StlMap< std::string, WireFormatFactory * >** (p. 2439), **decaf::util::StlMap< std::string, PrimitiveValueNode >** (p. 2439), **decaf::util::StlMap< std::string, cms::Queue * >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ProducerId >, ActiveMQProducer *, commands::ProducerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, CachedConsumer * >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ConsumerId >, ActiveMQConsumer *, commands::ConsumerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, TransportFactory * >** (p. 2439), **decaf::util::StlMap< int, Pointer< Command > >** (p. 2439), **decaf::util::StlMap< Pointer< commands::ConsumerId >, Dispatcher *, commands::ConsumerId::COMPARATOR >** (p. 2439), **decaf::util::StlMap< std::string, CachedProducer * >** (p. 2439), **decaf::util::StlMap< std::string, cms::Topic * >** (p. 2439), **decaf::util::StlQueue< Pointer< Transport > >** (p. 2445), **decaf::util::StlQueue< Pointer< MessageDispatch > >** (p. 2445), **decaf::util::StlQueue< Task >** (p. 2445), **decaf::util::StlQueue< Pointer< Command > >** (p. 2445), and **decaf::util::StlQueue< decaf::lang::Pointer< commands::MessageDispatch > >** (p. 2445).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**Synchronizable.h**

6.535 activemq::core::Synchronization Class Reference

Transacted Object **Synchronization** (p. 2516), used to sync the events of a Transaction with the items in the Transaction.

`#include <src/main/activemq/core/Synchronization.h>`

Public Member Functions

- virtual ~**Synchronization** ()
- virtual void **beforeEnd** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterCommit** ()=0 throw (exceptions::ActiveMQException)
- virtual void **afterRollback** ()=0 throw (exceptions::ActiveMQException)

6.535.1 Detailed Description

Transacted Object **Synchronization** (p. 2516), used to sync the events of a Transaction with the items in the Transaction.

6.535.2 Constructor & Destructor Documentation

6.535.2.1 virtual activemq::core::Synchronization::~Synchronization () [inline, virtual]

6.535.3 Member Function Documentation

6.535.3.1 virtual void activemq::core::Synchronization::afterCommit () throw (exceptions::ActiveMQException) [pure virtual]

6.535.3.2 virtual void activemq::core::Synchronization::afterRollback () throw (exceptions::ActiveMQException) [pure virtual]

6.535.3.3 virtual void activemq::core::Synchronization::beforeEnd () throw (exceptions::ActiveMQException) [pure virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/core/**Synchronization.h**

6.536 decaf::lang::System Class Reference

`#include <src/main/decaf/lang/System.h>`

Public Member Functions

- **System** ()
- virtual ~**System** ()

Static Public Member Functions

- static const **util::Map**< std::string, std::string > & **getenv** () throw (lang::Exception)

  Enumerates the system environment and returns a map of env variable names to the string values they hold.

- static std::string **getenv** (const std::string &name) throw (lang::Exception)

  Reads an environment value from the system and returns it as a string object.

- static void **unsetenv** (const std::string &name) throw (lang::Exception)

  Clears a set env value if one is set.

- static void **setenv** (const std::string &name, const std::string &value) throw (lang::Exception)

  Sets the specified system property to the value given.

- static long long **currentTimeMillis** ()
- static long long **nanoTime** ()

  Returns the current value of the most precise available system timer, in nanoseconds.

6.536.1 Constructor & Destructor Documentation

6.536.1.1 decaf::lang::System::System ()

6.536.1.2 virtual decaf::lang::System::~System () `[inline, virtual]`

6.536.2 Member Function Documentation

6.536.2.1 static long long decaf::lang::System::currentTimeMillis () `[static]`

Returns:

the current system time in Milliseconds

6.536.2.2 static std::string decaf::lang::System::getenv (const std::string & *name*) throw (lang::Exception) `[static]`

Reads an environment value from the system and returns it as a string object.

Parameters:

name - the env var to read

Returns:

a string with the value from the var or ""

Exceptions:

an **Exception** (p. 1268) if an error occurs while reading the Env.

6.536.2.3 static const util::Map<std::string, std::string>& decaf::lang::System::getenv () throw (lang::Exception) [static]

Enumerates the system environment and returns a map of env variable names to the string values they hold.

Returns:

A Map of all environment variables.

Exceptions:

Exception (p. ***1268)*** if an error occurs

6.536.2.4 static long long decaf::lang::System::nanoTime () [static]

Returns the current value of the most precise available system timer, in nanoseconds. This method can only be used to measure elapsed time and is not related to any other notion of system or wall-clock time. The value returned represents nanoseconds since some fixed but arbitrary time (perhaps in the future, so values may be negative). This method provides nanosecond precision, but not necessarily nanosecond accuracy. No guarantees are made about how frequently values change. Differences in successive calls that span greater than approximately 292 years (263 nanoseconds) will not accurately compute elapsed time due to numerical overflow.

For example, to measure how long some code takes to execute:

long long startTime = **System::nanoTime()** (p. 2518); // ... the code being measured ... long long estimatedTime = **System::nanoTime()** (p. 2518) - startTime;

Returns:

The current value of the system timer, in nanoseconds.

6.536.2.5 static void decaf::lang::System::setenv (const std::string & *name*, const std::string & *value*) throw (lang::Exception) [static]

Sets the specified system property to the value given.

Parameters:

name - name of the env val to set

value - value to assign to name

Exceptions:

***an* Exception** (p. 1268) if an error occurs

6.536.2.6 static void decaf::lang::System::unsetenv (const std::string & *name*) throw (lang::Exception) [static]

Clears a set env value if one is set.

Parameters:

name - the env var to clear

Exceptions:

***an* Exception** (p. 1268) if an error occurs while reading the Env.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**System.h**

6.537 activemq::threads::Task Class Reference

Represents a unit of work that requires one or more iterations to complete.

`#include <src/main/activemq/threads/Task.h>`Inheritance diagram for activemq::threads::Task:

Public Member Functions

- virtual ~**Task** ()
- virtual bool **iterate** ()=0

  Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

6.537.1 Detailed Description

Represents a unit of work that requires one or more iterations to complete.

Since:

3.0

6.537.2 Constructor & Destructor Documentation

6.537.2.1 virtual activemq::threads::Task::~Task () [inline, virtual]

6.537.3 Member Function Documentation

6.537.3.1 virtual bool activemq::threads::Task::iterate () [pure virtual]

Perform one iteration of work, returns true if the task needs to run again to complete or false to indicate that the task is now complete.

Returns:

true if the task should be run again or false if the task has completed and the runner should wait for a wakeup call.

Implemented in **activemq::core::ActiveMQSessionExecutor** (p. 371), **activemq::threads::CompositeTaskRunner** (p. 909), **activemq::transport::failover::BackupTransportPool** (p. 506), **activemq::transport::failover::CloseTransportsTask** (p. 842), and **activemq::transport::failover::FailoverTransport** (p. 1302).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**Task.h**

6.538 decaf::util::concurrent::TaskListener Class Reference

`#include <src/main/decaf/util/concurrent/TaskListener.h>`

Public Member Functions

- virtual ~**TaskListener** ()
- virtual void **onTaskComplete** (**lang::Runnable** *task)=0

  Called when a queued task has completed, the task that finished is passed along for user consumption.

- virtual void **onTaskException** (**lang::Runnable** *task, **lang::Exception** &ex)=0

  Called when a queued task has thrown an exception while being run.

6.538.1 Constructor & Destructor Documentation

6.538.1.1 virtual decaf::util::concurrent::TaskListener::~TaskListener () [inline, virtual]

6.538.2 Member Function Documentation

6.538.2.1 virtual void decaf::util::concurrent::TaskListener::onTaskComplete (lang::Runnable * *task*) [pure virtual]

Called when a queued task has completed, the task that finished is passed along for user consumption.

Parameters:

task Runnable Pointer to the task that finished

6.538.2.2 virtual void decaf::util::concurrent::TaskListener::onTaskException (lang::Runnable * *task*, lang::Exception & *ex*) [pure virtual]

Called when a queued task has thrown an exception while being run. The Callee should assume that this was an unrecoverable exeption and that this task is now defunct.

Parameters:

task Runnable Pointer to the task

ex The ActiveMQException that was thrown.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TaskListener.h**

6.539 activemq::threads::TaskRunner Class Reference

`#include <src/main/activemq/threads/TaskRunner.h>`Inheritance diagram for activemq::threads::TaskRunner:

Public Member Functions

- virtual ~**TaskRunner** ()
- virtual void **shutdown** (unsigned int timeout)=0

  Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

- virtual void **shutdown** ()=0

  Shutdown once the task has finished and the TaskRunner's thread has exited.

- virtual void **wakeup** ()=0

  *Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.*

6.539.1 Constructor & Destructor Documentation

6.539.1.1 virtual activemq::threads::TaskRunner::~TaskRunner () [inline, virtual]

6.539.2 Member Function Documentation

6.539.2.1 virtual void activemq::threads::TaskRunner::shutdown () [pure virtual]

Shutdown once the task has finished and the TaskRunner's thread has exited.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 909), and **activemq::threads::DedicatedTaskRunner** (p. 1184).

6.539.2.2 virtual void activemq::threads::TaskRunner::shutdown (unsigned int *timeout*) [pure virtual]

Shutdown after a timeout, does not guarantee that the task's iterate method has completed and the thread halted.

Parameters:

timeout - Time in Milliseconds to wait for the task to stop.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 910), and **activemq::threads::DedicatedTaskRunner** (p. 1184).

6.539.2.3 virtual void activemq::threads::TaskRunner::wakeup () [pure virtual]

Signal the **TaskRunner** (p. 2522) to wakeup and execute another iteration cycle on the task, the **Task** (p. 2520) instance will be run until its iterate method has returned false indicating it is done.

Implemented in **activemq::threads::CompositeTaskRunner** (p. 910), and **activemq::threads::DedicatedTaskRunner** (p. 1184).

The documentation for this class was generated from the following file:

- src/main/activemq/threads/**TaskRunner.h**

6.540 decaf::net::TcpSocket Class Reference

Platform-independent implementation of the socket interface.

`#include <src/main/decaf/net/TcpSocket.h>`Inheritance diagram for decaf::net::TcpSocket:

Public Member Functions

- **TcpSocket** () throw (SocketException)
  Construct a non-connected socket.
- **TcpSocket** (**SocketHandle** socketHandle)
  Construct a connected or bound socket based on given socket handle.
- virtual ~**TcpSocket** ()
  Releases the socket handle but not gracefully shut down the connection.
- **SocketHandle getSocketHandle** ()
  Gets the handle for the socket.
- void **connect** (const char *host, int port, int timeout) throw (SocketException)
  Connects to the specified destination.
- virtual void **connect** (const char *host, int port) throw (SocketException)
  Connects to the specified destination.
- virtual bool **isConnected** () const
  Indicates whether or not this socket is connected to a destination.
- virtual **io::InputStream** * **getInputStream** ()
  Gets the InputStream for this socket.
- virtual **io::OutputStream** * **getOutputStream** ()
  Gets the OutputStream for this socket.
- virtual int **getSoLinger** () const throw (SocketException)
  Gets the linger time.
- virtual void **setSoLinger** (int linger) throw (SocketException)
  Sets the linger time.
- virtual bool **getKeepAlive** () const throw (SocketException)
  Gets the keep alive flag.
- virtual void **setKeepAlive** (bool keepAlive) throw (SocketException)
  Enables/disables the keep alive flag.
- virtual int **getReceiveBufferSize** () const throw (SocketException)

Gets the receive buffer size.

- virtual void **setReceiveBufferSize** (int size) throw (SocketException)
  Sets the recieve buffer size.
- virtual bool **getReuseAddress** () const throw (SocketException)
  Gets the reuse address flag.
- virtual void **setReuseAddress** (bool reuse) throw (SocketException)
  Sets the reuse address flag.
- virtual int **getSendBufferSize** () const throw (SocketException)
  Gets the send buffer size.
- virtual void **setSendBufferSize** (int size) throw (SocketException)
  Sets the send buffer size.
- virtual int **getSoTimeout** () const throw (SocketException)
  Gets the timeout for socket operations.
- virtual void **setSoTimeout** (int timeout) throw (SocketException)
  Sets the timeout for socket operations.
- virtual void **close** () throw (lang::Exception)
  Closes this object and deallocates the appropriate resources.
- virtual bool **getTcpNoDelay** () const throw (lang::Exception)
  Gets the Status of the TCP_ NODELAY param for this socket as a Bool.
- virtual void **setTcpNoDelay** (bool value) throw (lang::Exception)
  Sets the Status of the TCP_ NODELAY param for this socket as a Bool.

Protected Member Functions

- void **checkResult** (apr_status_t value) const throw (SocketException)

6.540.1 Detailed Description

Platform-independent implementation of the socket interface.

6.540.2 Constructor & Destructor Documentation

6.540.2.1 decaf::net::TcpSocket::TcpSocket () throw (SocketException)

Construct a non-connected socket.

Exceptions:

SocketException *(*p. *2379)* thrown one windows if the static initialization call to WSAStartup was not successful.

6.540.2.2 decaf::net::TcpSocket::TcpSocket (SocketHandle *socketHandle*)

Construct a connected or bound socket based on given socket handle.

Parameters:

socketHandle a socket handle to wrap in the object

6.540.2.3 virtual decaf::net::TcpSocket::~TcpSocket () [virtual]

Releases the socket handle but not gracefully shut down the connection.

6.540.3 Member Function Documentation

6.540.3.1 void decaf::net::TcpSocket::checkResult (apr_status_t *value*) const throw (SocketException) [protected]

6.540.3.2 virtual void decaf::net::TcpSocket::close () throw (lang::Exception) [virtual]

Closes this object and deallocates the appropriate resources.

Exceptions:

Exception

Implements **decaf::io::Closeable** (p. 840).

6.540.3.3 virtual void decaf::net::TcpSocket::connect (const char * *host*, int *port*) throw (SocketException) [inline, virtual]

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

Exceptions:

SocketException *(*p. *2379)* Thrown if a failure occurred in the connect.

Implements **decaf::net::Socket** (p. 2372).

6.540.3.4 void decaf::net::TcpSocket::connect (const char * *host*, int *port*, int *timeout*) throw (SocketException)

Connects to the specified destination. Closes this socket if connected to another destination.

Parameters:

host The host of the server to connect to.

port The port of the server to connect to.

timeout of socket in microseconds

Exceptions:

SocketException *(*p. *2379)* Thrown if a failure occurred in the connect.

6.540.3.5 virtual io::InputStream* decaf::net::TcpSocket::getInputStream () [virtual]

Gets the InputStream for this socket.

Returns:

The InputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2373).

6.540.3.6 virtual bool decaf::net::TcpSocket::getKeepAlive () const throw (SocketException) [virtual]

Gets the keep alive flag.

Returns:

True if keep alive is enabled.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2373).

6.540.3.7 virtual io::OutputStream* decaf::net::TcpSocket::getOutputStream () [virtual]

Gets the OutputStream for this socket.

Returns:

the OutputStream for this socket. NULL if not connected.

Implements **decaf::net::Socket** (p. 2373).

6.540.3.8 virtual int decaf::net::TcpSocket::getReceiveBufferSize () const throw (SocketException) [virtual]

Gets the receive buffer size.

Returns:

the receive buffer size in bytes.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2373).

6.540.3.9 virtual bool decaf::net::TcpSocket::getReuseAddress () const throw (SocketException) [virtual]

Gets the reuse address flag.

Returns:

True if the address can be reused.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

6.540.3.10 virtual int decaf::net::TcpSocket::getSendBufferSize () const throw (SocketException) [virtual]

Gets the send buffer size.

Returns:

the size in bytes of the send buffer.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

6.540.3.11 SocketHandle decaf::net::TcpSocket::getSocketHandle () [inline]

Gets the handle for the socket.

Returns:

SocketHabler for this **Socket** (p. 2371), can be NULL

6.540.3.12 virtual int decaf::net::TcpSocket::getSoLinger () const throw (SocketException) [virtual]

Gets the linger time.

Returns:

The linger time in seconds.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2374).

6.540.3.13 virtual int decaf::net::TcpSocket::getSoTimeout () const throw (SocketException) [virtual]

Gets the timeout for socket operations.

Returns:

The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(* p. *2379)* Thrown if unable to retrieve the information.

Implements **decaf::net::Socket** (p. 2375).

6.540.3.14 virtual bool decaf::net::TcpSocket::getTcpNoDelay () const throw (lang::Exception) [virtual]

Gets the Status of the TCP_NODELAY param for this socket as a Bool.

Returns:

true if TCP_NODELAY is enabled

Exceptions:

Exception

6.540.3.15 virtual bool decaf::net::TcpSocket::isConnected () const [inline, virtual]

Indicates whether or not this socket is connected to a destination.

Returns:

true if connected

Implements **decaf::net::Socket** (p. 2375).

6.540.3.16 virtual void decaf::net::TcpSocket::setKeepAlive (bool *keepAlive*) throw (SocketException) [virtual]

Enables/disables the keep alive flag.

Parameters:

keepAlive If true, enables the flag.

Exceptions:

SocketException *(* p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2375).

6.540.3.17 virtual void decaf::net::TcpSocket::setReceiveBufferSize (int *size*) throw (SocketException) [virtual]

Sets the recieve buffer size.

Parameters:

size Number of bytes to set the receive buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2375).

6.540.3.18 virtual void decaf::net::TcpSocket::setReuseAddress (bool *reuse*) throw (SocketException) [virtual]

Sets the reuse address flag.

Parameters:

reuse If true, sets the flag.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

6.540.3.19 virtual void decaf::net::TcpSocket::setSendBufferSize (int *size*) throw (SocketException) [virtual]

Sets the send buffer size.

Parameters:

size The number of bytes to set the send buffer to.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

6.540.3.20 virtual void decaf::net::TcpSocket::setSoLinger (int *linger*) throw (SocketException) [virtual]

Sets the linger time.

Parameters:

linger The linger time in seconds. If 0, linger is off.

Exceptions:

SocketException *(*p. *2379)* if the operation fails.

Implements **decaf::net::Socket** (p. 2376).

6.540.3.21 virtual void decaf::net::TcpSocket::setSoTimeout (int *timeout*) throw (SocketException) [virtual]

Sets the timeout for socket operations.

Parameters:

timeout The timeout in milliseconds for socket operations.

Exceptions:

SocketException *(*p. *2379)* Thrown if unable to set the information.

Implements **decaf::net::Socket** (p. 2377).

6.540.3.22 virtual void decaf::net::TcpSocket::setTcpNoDelay (bool *value*) throw (lang::Exception) [virtual]

Sets the Status of the TCP_NODELAY param for this socket as a Bool.

Parameters:

value - true if TCP_NODELAY is to be enabled

Exceptions:

Exception

The documentation for this class was generated from the following file:

- src/main/decaf/net/**TcpSocket.h**

6.541 activemq::transport::tcp::TcpTransport Class Reference

Implements a TCP/IP based **transport** (p. 67) filter, this **transport** (p. 67) is meant to wrap an instance of an **IOTransport** (p. 1480).

`#include <src/main/activemq/transport/tcp/TcpTransport.h>`Inheritance diagram for activemq::transport::tcp::TcpTransport:

Public Member Functions

- **TcpTransport** (const **decaf::util::Properties** &properties, const **Pointer< Transport > &next**)

  Constructor.

- **TcpTransport** (const **decaf::net::URI** &uri, const **decaf::util::Properties** &properties, const **Pointer< Transport > &next**)

  Constructor.

- virtual ~**TcpTransport** ()
- virtual void **close** () throw (cms::CMSException)

  Delegates to the superclass and then closes the socket.

- virtual bool **isFaultTolerant** () const

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

6.541.1 Detailed Description

Implements a TCP/IP based **transport** (p. 67) filter, this **transport** (p. 67) is meant to wrap an instance of an **IOTransport** (p. 1480). The lower level **transport** (p. 67) should take care of managing stream reads and writes.

6.541.2 Constructor & Destructor Documentation

6.541.2.1 activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::util::Properties & *properties*, const Pointer< Transport > & *next*)

Constructor.

Parameters:

properties the configuration properties for this **transport** (p. 67)

next the next **transport** (p. 67) in the chain

6.541.2.2 activemq::transport::tcp::TcpTransport::TcpTransport (const decaf::net::URI & *uri*, const decaf::util::Properties & *properties*, const Pointer< Transport > & *next*)

Constructor.

Parameters:

uri - The URI containing the host to connect to.

properties the configuration properties for this **transport** (p. 67)

next the next **transport** (p. 67) in the chain

6.541.2.3 virtual activemq::transport::tcp::TcpTransport::~TcpTransport () [virtual]

6.541.3 Member Function Documentation

6.541.3.1 virtual void activemq::transport::tcp::TcpTransport::close () throw (cms::CMSException) [virtual]

Delegates to the superclass and then closes the socket.

Exceptions:

CMSException if errors occur.

Reimplemented from **activemq::transport::TransportFilter** (p. 2618).

6.541.3.2 virtual bool activemq::transport::tcp::TcpTransport::isClosed () const [inline, virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Reimplemented from **activemq::transport::TransportFilter** (p. 2619).

6.541.3.3 virtual bool activemq::transport::tcp::TcpTransport::isConnected () const [inline, virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Reimplemented from **activemq::transport::TransportFilter** (p. 2619).

6.541.3.4 virtual bool activemq::transport::tcp::TcpTransport::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Reimplemented from **activemq::transport::TransportFilter** (p. 2619).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransport.h**

6.542 activemq::transport::tcp::TcpTransportFactory Class Reference

Factory Responsible for creating the **TcpTransport** (p. 2532).

`#include <src/main/activemq/transport/tcp/TcpTransportFactory.h>`Inheritance diagram for activemq::transport::tcp::TcpTransportFactory:

Public Member Functions

- virtual ~**TcpTransportFactory** ()
- virtual **Pointer**< **Transport** > **create** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.*

- virtual **Pointer**< **Transport** > **createComposite** (const **decaf::net::URI** &location) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

Protected Member Functions

- virtual **Pointer**< **Transport** > **doCreateComposite** (const **decaf::net::URI** &location, const **Pointer**< **wireformat::WireFormat** > &wireFormat, const **decaf::util::Properties** &properties) throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

6.542.1 Detailed Description

Factory Responsible for creating the **TcpTransport** (p. 2532).

6.542.2 Constructor & Destructor Documentation

6.542.2.1 virtual activemq::transport::tcp::TcpTransportFactory::~TcpTransportFactory () [inline, virtual]

6.542.3 Member Function Documentation

6.542.3.1 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::create (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2614).

6.542.3.2 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::createComposite (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implements **activemq::transport::TransportFactory** (p. 2615).

6.542.3.3 virtual Pointer<Transport> activemq::transport::tcp::TcpTransportFactory::doCreateComposite (const decaf::net::URI & *location*, const Pointer< wireformat::WireFormat > & *wireFormat*, const decaf::util::Properties & *properties*) throw (exceptions::ActiveMQException) [protected, virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to.

wireFormat - the assigned WireFormat for the new **Transport** (p. 2608).

properties - Properties to apply to the **transport** (p. 67).

Returns:

new Pointer to a **TcpTransport** (p. 2532).

Exceptions:

ActiveMQexception if an error occurs

The documentation for this class was generated from the following file:

- src/main/activemq/transport/tcp/**TcpTransportFactory.h**

6.543 cms::TemporaryQueue Class Reference

Defines a Temporary **Queue** (p. 2157) based **Destination** (p. 1190).

`#include <src/main/cms/TemporaryQueue.h>`Inheritance diagram for cms::TemporaryQueue:

Public Member Functions

- virtual ~**TemporaryQueue** ()
- virtual std::string **getQueueName** () const =0 throw (CMSException)

  Gets the name of this queue.

- virtual void **destroy** ()=0 throw (CMSException)

  *Destroy's the Temporary **Destination** (p. 1190) at the Provider.*

6.543.1 Detailed Description

Defines a Temporary **Queue** (p. 2157) based **Destination** (p. 1190). A **TemporaryQueue** (p. 2538) is a special type of **Queue** (p. 2157) **Destination** (p. 1190) that can only be consumed from the **Connection** (p. 941) which created it. TemporaryQueues are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryQueue** (p. 2538) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 941) that created it.

Since:

1.0

6.543.2 Constructor & Destructor Documentation

6.543.2.1 virtual cms::TemporaryQueue::~TemporaryQueue () `[inline, virtual]`

6.543.3 Member Function Documentation

6.543.3.1 virtual void cms::TemporaryQueue::destroy () throw (CMSException) `[pure virtual]`

Destroy's the Temporary **Destination** (p. 1190) at the Provider.

Exceptions:

***CMSException** (p. **850**)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 417).

6.543.3.2 virtual std::string cms::TemporaryQueue::getQueueName () const throw (CMSException) [pure virtual]

Gets the name of this queue.

Returns:

The queue name.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempQueue** (p. 418).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryQueue.h**

6.544 cms::TemporaryTopic Class Reference

Defines a Temporary **Topic** (p. 2571) based **Destination** (p. 1190).

`#include <src/main/cms/TemporaryTopic.h>`Inheritance diagram for cms::TemporaryTopic:

Public Member Functions

- virtual ~**TemporaryTopic** ()
- virtual std::string **getTopicName** () const =0 throw (CMSException)

  Gets the name of this topic.

- virtual void **destroy** ()=0 throw (CMSException)

  *Destroy's the Temporary **Destination** (p. 1190) at the Provider.*

6.544.1 Detailed Description

Defines a Temporary **Topic** (p. 2571) based **Destination** (p. 1190). A **TemporaryTopic** (p. 2540) is a special type of **Topic** (p. 2571) **Destination** (p. 1190) that can only be consumed from the **Connection** (p. 941) which created it. TemporaryTopics are most commonly used as the reply to address for Message's that implement the request response pattern.

A **TemporaryTopic** (p. 2540) is guaranteed to exist at the Provider only for the lifetime of the **Connection** (p. 941) that created it.

Since:

1.0

6.544.2 Constructor & Destructor Documentation

6.544.2.1 virtual cms::TemporaryTopic::~TemporaryTopic () [inline, virtual]

6.544.3 Member Function Documentation

6.544.3.1 virtual void cms::TemporaryTopic::destroy () throw (CMSException) [pure virtual]

Destroy's the Temporary **Destination** (p. 1190) at the Provider.

Exceptions:

***CMSException** (p. 850)*

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 434).

6.544.3.2 virtual std::string cms::TemporaryTopic::getTopicName () const throw (CMSException) [pure virtual]

Gets the name of this topic.

Returns:

The topic name.

Exceptions:

CMSException *(*p. ***850)*** - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTempTopic** (p. 435).

The documentation for this class was generated from the following file:

- src/main/cms/**TemporaryTopic.h**

6.545 cms::TextMessage Class Reference

Interface for a text message.

`#include <src/main/cms/TextMessage.h>`Inheritance diagram for cms::TextMessage:

Public Member Functions

- virtual ~**TextMessage** ()
- virtual std::string **getText** () const =0 throw (CMSException)

  Gets the message character buffer.

- virtual void **setText** (const char *msg)=0 throw (CMSException)

  Sets the message contents, does not take ownership of the passed char, but copies it instead.*

- virtual void **setText** (const std::string &msg)=0 throw (CMSException)

  Sets the message contents.

6.545.1 Detailed Description

Interface for a text message. A **TextMessage** (p. 2542) can contain any Text based pay load such as an XML Document or other Text based document.

Like all Messages, a **TextMessage** (p. 2542) is received in Read-Only mode, any attempt to write to the **Message** (p. 1753) will result in a MessageNotWritableException being thrown until the `clearBody` method is called which will erase the contents and place the message back in a read / write mode.

Since:

1.0

6.545.2 Constructor & Destructor Documentation

6.545.2.1 virtual cms::TextMessage::~TextMessage () [inline, virtual]

6.545.3 Member Function Documentation

6.545.3.1 virtual std::string cms::TextMessage::getText () const throw (CMSException) [pure virtual]

Gets the message character buffer.

Returns:

The message character buffer.

Exceptions:

CMSException *(*p. ***850****)* - if an internal error occurs.

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 451).

6.545.3.2 virtual void cms::TextMessage::setText (const std::string & *msg*) throw (CMSException) [pure virtual]

Sets the message contents.

Parameters:

msg The message buffer.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 451).

6.545.3.3 virtual void cms::TextMessage::setText (const char * *msg*) throw (CMSException) [pure virtual]

Sets the message contents, does not take ownership of the passed char*, but copies it instead.

Parameters:

msg The message buffer.

Exceptions:

CMSException (p. ***850***) - if an internal error occurs.

MessageNotWriteableException (p. ***1877***) - if the message is in read-only mode..

Implemented in **activemq::commands::ActiveMQTextMessage** (p. 451).

The documentation for this class was generated from the following file:

- src/main/cms/**TextMessage.h**

6.546 decaf::lang::Thread Class Reference

Basic thread class - mimics the Java **Thread** (p. 2544).

`#include <src/main/decaf/lang/Thread.h>`Inheritance diagram for decaf::lang::Thread:

Public Member Functions

- **Thread** ()
  default Constructor
- **Thread** (**Runnable** *task)
  Constructor.
- virtual ~**Thread** ()
- virtual void **start** () throw (Exception)
  Creates a system thread and starts it in a joinable mode.
- virtual void **join** () throw (Exception)
  Wait til the thread exits.
- virtual void **run** ()
  Default implementation of the run method - does nothing.

Static Public Member Functions

- static void **sleep** (int millisecs)
  Halts execution of the calling thread for a specified no of millisec.
- static void **yield** ()
  Causes the currently executing thread object to temporarily pause and allow other threads to execute.
- static unsigned long **getId** ()
  *Obtains the **Thread** (p. 2544) Id of the current thread.*

6.546.1 Detailed Description

Basic thread class - mimics the Java **Thread** (p. 2544). Derived classes may implement the run method, or this class can be used as is with a provided **Runnable** (p. 2256) delegate.

6.546.2 Constructor & Destructor Documentation

6.546.2.1 decaf::lang::Thread::Thread ()

default Constructor

6.546.2.2 decaf::lang::Thread::Thread (Runnable * *task*)

Constructor.

Parameters:

task the **Runnable** (p. 2256) that this thread manages

6.546.2.3 virtual decaf::lang::Thread::~Thread () [virtual]

6.546.3 Member Function Documentation

6.546.3.1 static unsigned long decaf::lang::Thread::getId () [static]

Obtains the **Thread** (p. 2544) Id of the current thread.

Returns:

Thread (p. 2544) Id

6.546.3.2 virtual void decaf::lang::Thread::join () throw (Exception) [virtual]

Wait til the thread exits. This is when the **run()** (p. 2545) method has returned or has thrown an exception.

6.546.3.3 virtual void decaf::lang::Thread::run () [inline, virtual]

Default implementation of the run method - does nothing.

Implements **decaf::lang::Runnable** (p. 2256).

Reimplemented in **activemq::transport::mock::InternalCommandListener** (p. 1458), and **decaf::util::concurrent::PooledThread** (p. 2033).

6.546.3.4 static void decaf::lang::Thread::sleep (int *millisecs*) [static]

Halts execution of the calling thread for a specified no of millisec. Note that this method is a static method that applies to the calling thread and not to the thread object.

Parameters:

millisecs time in milliseconds to sleep

6.546.3.5 virtual void decaf::lang::Thread::start () throw (Exception) [virtual]

Creates a system thread and starts it in a joinable mode. Upon creation, the **run()** (p. 2545) method of either this object or the provided **Runnable** (p. 2256) object will be invoked in the context of this thread.

Exceptions:

runtime_error is thrown if the system could not start the thread.

6.546.3.6 static void decaf::lang::Thread::yield () [static]

Causes the currently executing thread object to temporarily pause and allow other threads to execute.

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Thread.h**

6.547 decaf::util::concurrent::ThreadFactory Class Reference

public interface **ThreadFactory** (p. 2547)

`#include <src/main/decaf/util/concurrent/ThreadFactory.h>`

Public Member Functions

- virtual ~**ThreadFactory** ()
- virtual Thread * **newThread** (Runnable *r)=0

  Constructs a new Thread.

6.547.1 Detailed Description

public interface **ThreadFactory** (p. 2547) An object that creates new threads on demand. Using thread factories removes hardwiring of calls to new Thread, enabling applications to use special thread subclasses, priorities, etc.

The simplest implementation of this interface is just:

class SimpleThreadFactory : public **ThreadFactory** (p. 2547) { public: Thread* **newThread(Runnable* r)** (p. 2547) { return new Thread(r); } }

The Executors.defaultThreadFactory() method provides a more useful simple implementation, that sets the created thread context to known values before returning it.

Since:

1.0

6.547.2 Constructor & Destructor Documentation

6.547.2.1 virtual decaf::util::concurrent::ThreadFactory::~ThreadFactory () [inline, virtual]

6.547.3 Member Function Documentation

6.547.3.1 virtual Thread* decaf::util::concurrent::ThreadFactory::newThread (Runnable * r) [pure virtual]

Constructs a new Thread. Implementations may also initialize priority, name, daemon status, ThreadGroup, etc.

Parameters:

r - a runnable to be executed by new thread instance

Returns:

constructed thread, or null if the request to create a thread is rejected

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadFactory.h**

6.548 decaf::util::concurrent::ThreadPool Class Reference

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

`#include <src/main/decaf/util/concurrent/ThreadPool.h>`Inheritance diagram for decaf::util::concurrent::ThreadPool:

Public Types

- typedef std::pair< **lang::Runnable** *, **TaskListener** * > **Task**

Public Member Functions

- **ThreadPool** ()
- virtual ~**ThreadPool** ()
- virtual void **queueTask** (**Task** task) throw (lang::Exception)

  Queue *(p. 2154) a task to be completed by one of the Pooled Threads.*

- virtual **Task deQueueTask** () throw (lang::Exception)

  DeQueue a task to be completed by one of the Pooled Threads.

- virtual std::size_t **getPoolSize** () const

  Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

- virtual std::size_t **getBacklog** () const

  Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

- virtual void **reserve** (std::size_t size)

  Ensures that there is at least the specified number of Threads allocated to the pool.

- virtual std::size_t **getMaxThreads** () const

  Get the Max Number of Threads this Pool can contain.

- virtual void **setMaxThreads** (std::size_t maxThreads)

  Sets the Max number of threads this pool can contian.

- virtual std::size_t **getBlockSize** () const

  Gets the Max number of threads that can be allocated at a time when new threads are needed.

- virtual void **setBlockSize** (std::size_t blockSize)

  Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

- virtual std::size_t **getFreeThreadCount** () const

  Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable.

- virtual void **onTaskStarted** (**PooledThread** *thread)

  Called by a pooled thread when it is about to begin executing a new task.

- virtual void **onTaskCompleted** (**PooledThread** *thread)

  Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

- virtual void **onTaskException** (**PooledThread** *thread, **lang::Exception** &ex)

  *Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2032) is now no longer running.*

Static Public Member Functions

- static **ThreadPool** * **getInstance** ()

  Return the one and only Thread Pool instance.

Static Public Attributes

- static const size_t **DEFAULT_MAX_POOL_SIZE** = 10
- static const size_t **DEFAULT_MAX_BLOCK_SIZE** = 3

6.548.1 Detailed Description

Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks. The Thread Poll has max size that it will grow to. The thread pool allocates threads in blocks. When there are no waiting worker threads and a task is queued then a new batch is allocated. The user can specify the size of the blocks, otherwise a default value is used.

When the user queues a task they must also queue a listener to be notified when the task has completed, this provides the user with a mechanism to know when a task object can be freed.

To have the Thread Pool perform a task, the user enqueue's an object that implements the `Runnable` interface and one of the worker threads will executing it in its thread context.

6.548.2 Member Typedef Documentation

6.548.2.1 typedef std::pair<lang::Runnable*, TaskListener*> decaf::util::concurrent::ThreadPool::Task

6.548.3 Constructor & Destructor Documentation

6.548.3.1 decaf::util::concurrent::ThreadPool::ThreadPool ()

6.548.3.2 virtual decaf::util::concurrent::ThreadPool::~ThreadPool () [virtual]

6.548.4 Member Function Documentation

6.548.4.1 virtual Task decaf::util::concurrent::ThreadPool::deQueueTask () throw (lang::Exception) [virtual]

DeQueue a task to be completed by one of the Pooled Threads. A caller of this method will block until there is something in the tasks queue, therefore care must be taken when calling this function. Normally clients of **ThreadPool** (p. 2549) don't use this, only the PooledThread (p. 2032) objects owned by this **ThreadPool** (p. 2549).

Returns:

object that derives from Runnable

Exceptions:

ActiveMQException

6.548.4.2 virtual std::size_t decaf::util::concurrent::ThreadPool::getBacklog () const [inline, virtual]

Returns the current backlog of items in the tasks queue, this is how much work is still waiting to get done.

Returns:

number of outstanding tasks.

6.548.4.3 virtual std::size_t decaf::util::concurrent::ThreadPool::getBlockSize () const [inline, virtual]

Gets the Max number of threads that can be allocated at a time when new threads are needed.

Returns:

max Thread Block Size

6.548.4.4 virtual std::size_t decaf::util::concurrent::ThreadPool::getFreeThreadCount () const [inline, virtual]

Returns the current number of available threads in the pool, threads that are performing a user task are considered unavailable. This value could change immediately after calling as Threads

could finish right after and be available again. This is informational only.

Returns:

total free threads

6.548.4.5 static ThreadPool* decaf::util::concurrent::ThreadPool::getInstance () [static]

Return the one and only Thread Pool instance.

Returns:

The Thread Pool Pointer

6.548.4.6 virtual std::size_t decaf::util::concurrent::ThreadPool::getMaxThreads () const [inline, virtual]

Get the Max Number of Threads this Pool can contain.

Returns:

max size

6.548.4.7 virtual std::size_t decaf::util::concurrent::ThreadPool::getPoolSize () const [inline, virtual]

Returns the current number of Threads in the Pool, this is how many there are now, not how many are active or the max number that might exist.

Returns:

integer number of threads in existence.

6.548.4.8 virtual void decaf::util::concurrent::ThreadPool::onTaskCompleted (PooledThread * *thread*) [virtual]

Called by a pooled thread when it has completed a task and is going back to waiting for another task to run, this will increment the free threads counter.

Parameters:

thread Pointer the the Pooled Thread that is making this call.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2035).

6.548.4.9 virtual void decaf::util::concurrent::ThreadPool::onTaskException (PooledThread * *thread*, lang::Exception & *ex*) [virtual]

Called by a pooled thread when it has encountered an exception while running a user task, after receiving this notification the callee should assume that the **PooledThread** (p. 2032) is now no longer running.

Parameters:

thread Pointer to the Pooled Thread that is making this call

ex The Exception that occurred.

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2036).

6.548.4.10 virtual void decaf::util::concurrent::ThreadPool::onTaskStarted (PooledThread * *thread*) [virtual]

Called by a pooled thread when it is about to begin executing a new task. This will decrement the available threads counter so that this object knows when there are no more free threads and must create new ones.

Parameters:

thread Pointer to the Pooled Thread that is making this call

Implements **decaf::util::concurrent::PooledThreadListener** (p. 2036).

6.548.4.11 virtual void decaf::util::concurrent::ThreadPool::queueTask (Task *task*) throw (lang::Exception) [virtual]

Queue (p. 2154) a task to be completed by one of the Pooled Threads. tasks are serviced as soon as a `PooledThread` (p. 2032) is available to run it.

Parameters:

task object that derives from Runnable

Exceptions:

ActiveMQException

6.548.4.12 virtual void decaf::util::concurrent::ThreadPool::reserve (std::size_t *size*) [virtual]

Ensures that there is at least the specified number of Threads allocated to the pool. If the size is greater than the MAX number of threads in the pool, then only MAX threads are reservved. If the size is smaller than the number of threads currently in the pool, than nothing is done.

Parameters:

size the number of threads to reserve.

6.548.4.13 virtual void decaf::util::concurrent::ThreadPool::setBlockSize (std::size_t *blockSize*) [virtual]

Sets the Max number of Threads that can be allocated at a time when the Thread Pool determines that more Threads are needed.

Parameters:

blockSize Max Thread Block Size

6.548.4.14 virtual void decaf::util::concurrent::ThreadPool::setMaxThreads (std::size_t *maxThreads*) [virtual]

Sets the Max number of threads this pool can contian. if this value is smaller than the current size of the pool nothing is done.

Parameters:

maxThreads total number of threads that can be pooled

6.548.5 Field Documentation

6.548.5.1 const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_BLOCK_SIZE = 3 [static]

6.548.5.2 const size_t decaf::util::concurrent::ThreadPool::DEFAULT_MAX_POOL_SIZE = 10 [static]

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**ThreadPool.h**

6.549 decaf::lang::Throwable Class Reference

This class represents an error that has occurred.

`#include <src/main/decaf/lang/Throwable.h>`Inheritance diagram for decaf::lang::Throwable:

Public Member Functions

- **Throwable** () throw ()
- virtual ~**Throwable** () throw ()
- virtual std::string **getMessage** () const =0

  Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.

- virtual const std::exception * **getCause** () const =0

  Gets the exception that caused this one to be thrown, this allows for chaining of ***exceptions*** *(p. 104) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.*

- virtual void **initCause** (const std::exception *cause)=0

  Initializes the contained cause exception with the one given.

- virtual void **setMark** (const char *file, const int lineNumber)=0

  Adds a file/line number to the stack trace.

- virtual **Throwable** * **clone** () const =0

  Clones this exception.

- virtual std::vector< std::pair< std::string, int > > **getStackTrace** () const =0

  Provides the stack trace for every point where this exception was caught, marked, and rethrown.

- virtual void **printStackTrace** () const =0

  Prints the stack trace to std::err.

- virtual void **printStackTrace** (std::ostream &stream) const =0

  Prints the stack trace to the given output stream.

- virtual std::string **getStackTraceString** () const =0

  Gets the stack trace as one contiguous string.

6.549.1 Detailed Description

This class represents an error that has occurred. All Exceptions in the Decaf library should extend from this or from the **Exception** (p. 1268) class in order to ensure that all Decaf Exceptions are interchangeable with the std::exception class.

Throwable (p. 2555) can wrap another **Throwable** (p. 2555) as the cause if the error being thrown. The user can inspect the cause by calling `getCause`, the pointer returned is the property of the **Throwable** (p. 2555) instance and will be deleted when it is deleted or goes out of scope.

Since:

1.0

6.549.2 Constructor & Destructor Documentation

6.549.2.1 decaf::lang::Throwable::Throwable () throw () [inline]

6.549.2.2 virtual decaf::lang::Throwable::~Throwable () throw () [inline, virtual]

6.549.3 Member Function Documentation

6.549.3.1 virtual Throwable* decaf::lang::Throwable::clone () const [pure virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

Copy of this **Exception** (p. 1268) object

Implemented in **activemq::exceptions::ActiveMQException** (p. 253), **activemq::exceptions::BrokerException** (p. 590), **decaf::io::EOFException** (p. 1267), **decaf::io::InterruptedIOException** (p. 1464), **decaf::io::IOException** (p. 1479), **decaf::io::UTFDataFormatException** (p. 2685), **decaf::lang::Exception** (p. 1271), **decaf::lang::exceptions::ClassCastException** (p. 837), **decaf::lang::exceptions::IllegalArgumentException** (p. 1395), **decaf::lang::exceptions::IllegalMonitorStateException** (p. 1398), **decaf::lang::exceptions::IllegalStateException** (p. 1402), **decaf::lang::exceptions::IndexOutOfBoundsException** (p. 1405), **decaf::lang::exceptions::InterruptedException** (p. 1461), **decaf::lang::exceptions::InvalidStateException** (p. 1476), **decaf::lang::exceptions::NoSuchElementException** (p. 1947), **decaf::lang::exceptions::NullPointerException** (p. 1953), **decaf::lang::exceptions::NumberFormatException** (p. 1959), **decaf::lang::exceptions::RuntimeException** (p. 2261), **decaf::lang::exceptions::UnsupportedOperationException** (p. 2637), **decaf::net::BindException** (p. 565), **decaf::net::ConnectException** (p. 940), **decaf::net::HttpRetryException** (p. 1392), **decaf::net::MalformedURLException** (p. 1688), **decaf::net::NoRouteToHostException** (p. 1941), **decaf::net::PortUnreachableException** (p. 2039), **decaf::net::ProtocolException** (p. 2152), **decaf::net::SocketException** (p. 2380), **decaf::net::SocketTimeoutException** (p. 2397), **decaf::net::UnknownHostException** (p. 2631), **decaf::net::UnknownServiceException** (p. 2634), **decaf::net::URISyntaxException** (p. 2667), **decaf::nio::BufferOverflowException** (p. 660), **decaf::nio::BufferUnderflowException** (p. 663), **decaf::nio::InvalidMarkException** (p. 1472), **decaf::nio::ReadOnlyBufferException** (p. 2169), **decaf::util::concurrent::BrokenBarrierException** (p. 585), **decaf::util::concurrent::CancellationException** (p. 786), **decaf::util::concurrent::ExecutionException** (p. 1293), **de-**

caf::util::concurrent::RejectedExecutionException (p. 2176), and **decaf::util::concurrent::TimeoutException** (p. 2561).

6.549.3.2 virtual const std::exception* decaf::lang::Throwable::getCause () const [pure virtual]

Gets the exception that caused this one to be thrown, this allows for chaining of **exceptions** (p. 104) in the case of a method that throws only a particular exception but wishes to allow for the real causal exception to be passed only in case the caller knows about that type of exception and wishes to respond to it.

Returns:

a const pointer reference to the causal exception, if there was no cause associated with this exception then NULL is returned.

Implemented in **decaf::lang::Exception** (p. 1272).

6.549.3.3 virtual std::string decaf::lang::Throwable::getMessage () const [pure virtual]

Gets the cause of the error, if no message was provided to the instance of this interface but a cause was then the value cause.getMessage is then returned.

Returns:

string errors message

Implemented in **decaf::lang::Exception** (p. 1272).

6.549.3.4 virtual std::vector< std::pair< std::string, int> > decaf::lang::Throwable::getStackTrace () const [pure virtual]

Provides the stack trace for every point where this exception was caught, marked, and rethrown.

Returns:

vector containing stack trace strings

Implemented in **decaf::lang::Exception** (p. 1272).

6.549.3.5 virtual std::string decaf::lang::Throwable::getStackTraceString () const [pure virtual]

Gets the stack trace as one contiguous string.

Returns:

string with formatted stack trace data

Implemented in **decaf::lang::Exception** (p. 1272).

6.549.3.6 virtual void decaf::lang::Throwable::initCause (const std::exception * *cause*) [pure virtual]

Initializes the contained cause exception with the one given. A copy is made to avoid ownership issues.

Parameters:

cause The exception that was the cause of this one.

Implemented in **decaf::lang::Exception** (p. 1272).

6.549.3.7 virtual void decaf::lang::Throwable::printStackTrace (std::ostream & *stream*) const [pure virtual]

Prints the stack trace to the given output stream.

Parameters:

stream the target output stream.

Implemented in **decaf::lang::Exception** (p. 1273).

6.549.3.8 virtual void decaf::lang::Throwable::printStackTrace () const [pure virtual]

Prints the stack trace to std::err.

Implemented in **decaf::lang::Exception** (p. 1273).

6.549.3.9 virtual void decaf::lang::Throwable::setMark (const char * *file*, const int *lineNumber*) [pure virtual]

Adds a file/line number to the stack trace.

Parameters:

file The name of the file calling this method (use __FILE__).

lineNumber The line number in the calling file (use __LINE__).

Implemented in **decaf::lang::Exception** (p. 1273).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/**Throwable.h**

6.550 decaf::util::concurrent::TimeoutException Class Reference

`#include <src/main/decaf/util/concurrent/TimeoutException.h>`Inheritance diagram for decaf::util::concurrent::TimeoutException:

Public Member Functions

- **TimeoutException** () throw ()
  Default Constructor.
- **TimeoutException** (const **decaf::lang::Exception** &ex) throw ()
  Conversion Constructor from some other Exception.
- **TimeoutException** (const **TimeoutException** &ex) throw ()
  Copy Constructor.
- **TimeoutException** (const std::exception ∗**cause**) throw ()
  Constructor.
- **TimeoutException** (const char ∗file, const int lineNumber, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **TimeoutException** (const char ∗file, const int lineNumber, const std::exception ∗**cause**, const char ∗msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **TimeoutException** ∗ **clone** () const
  Clones this exception.
- virtual ∼**TimeoutException** () throw ()

6.550.1 Constructor & Destructor Documentation

6.550.1.1 decaf::util::concurrent::TimeoutException::TimeoutException () throw () [inline]

Default Constructor.

6.550.1.2 decaf::util::concurrent::TimeoutException::TimeoutException (const decaf::lang::Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.550.1.3 decaf::util::concurrent::TimeoutException::TimeoutException (const TimeoutException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex The exception to copy from.

References decaf::lang::Exception::Exception().

6.550.1.4 decaf::util::concurrent::TimeoutException::TimeoutException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.550.1.5 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The string message to report

... list of primitives that are formatted into the message

6.550.1.6 decaf::util::concurrent::TimeoutException::TimeoutException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The string message to report

... list of primitives that are formatted into the message

6.550.1.7 virtual decaf::util::concurrent::TimeoutException::~TimeoutException () throw () [inline, virtual]

6.550.2 Member Function Documentation

6.550.2.1 virtual TimeoutException* decaf::util::concurrent::TimeoutException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new **TimeoutException** (p. 2559) that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TimeoutException.h**

6.551 decaf::util::concurrent::TimeUnit Class Reference

A **TimeUnit** (p. 2562) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.

`#include <src/main/decaf/util/concurrent/TimeUnit.h>`Inheritance diagram for decaf::util::concurrent::TimeUnit:

Public Member Functions

- virtual ~**TimeUnit** ()
- long long **convert** (long long sourceDuration, const **TimeUnit** &sourceUnit) const

  Convert the given time duration in the given unit to this unit.

- long long **toNanos** (long long duration) const

  Equivalent to `NANOSECONDS.convert(duration, this)`.

- long long **toMicros** (long long duration) const

  Equivalent to `MICROSECONDS.convert(duration, this)`.

- long long **toMillis** (long long duration) const

  Equivalent to `MILLISECONDS.convert(duration, this)`.

- long long **toSeconds** (long long duration) const

  Equivalent to `SECONDS.convert(duration, this)`.

- long long **toMinutes** (long long duration) const

  Equivalent to `MINUTES.convert(duration, this)`.

- long long **toHours** (long long duration) const

  Equivalent to `HOURS.convert(duration, this)`.

- long long **toDays** (long long duration) const

  Equivalent to `DAYS.convert(duration, this)`.

- void **timedWait** (**Synchronizable** *obj, long long timeout) const

  Perform a timed `Object.wait` using this time unit.

- void **sleep** (long long timeout) const

  Perform a timed `Thread.join` using this time unit.

- virtual std::string **toString** () const

  *Converts the **TimeUnit** (p. 2562) type to the Name of the **TimeUnit** (p. 2562).*

- virtual int **compareTo** (const **TimeUnit** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **TimeUnit** &value) const

- virtual bool **operator==** (const **TimeUnit** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **TimeUnit** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

Static Public Member Functions

- static const **TimeUnit** & **valueOf** (const std::string &name) throw (decaf::lang::exceptions::IllegalArgumentException)

  *Returns the **TimeUnit** (p. 2562) constant of this type with the specified name.*

Static Public Attributes

- static const **TimeUnit NANOSECONDS**

  *The Actual **TimeUnit** (p. 2562) enumerations.*

- static const **TimeUnit MICROSECONDS**
- static const **TimeUnit MILLISECONDS**
- static const **TimeUnit SECONDS**
- static const **TimeUnit MINUTES**
- static const **TimeUnit HOURS**
- static const **TimeUnit DAYS**
- static const **TimeUnit** *const **values** [ ]

  *The An Array of **TimeUnit** (p. 2562) Instances.*

Protected Member Functions

- **TimeUnit** (int index, const std::string &name)

  Hidden Constructor, this class can not be instantiated directly.

6.551.1 Detailed Description

A **TimeUnit** (p. 2562) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A **TimeUnit** (p. 2562) does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A **TimeUnit** (p. 2562) is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the lock is not available:

Lock (p. 1616) lock = ...; if (lock.tryLock(50, **TimeUnit.MILLISECONDS** (p. 2570))) ...

while this code will timeout in 50 seconds:

Lock (p. 1616) lock = ...; if (lock.tryLock(50, **TimeUnit.SECONDS** (p. 2570))) ...

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given **TimeUnit** (p. 2562).

6.551.2 Constructor & Destructor Documentation

6.551.2.1 decaf::util::concurrent::TimeUnit::TimeUnit (int *index*, const std::string & *name*) [protected]

Hidden Constructor, this class can not be instantiated directly.

Parameters:

index - Index into the Time Unit set.

name - Name of the unit type being represented.

6.551.2.2 virtual decaf::util::concurrent::TimeUnit::~TimeUnit () [inline, virtual]

6.551.3 Member Function Documentation

6.551.3.1 virtual int decaf::util::concurrent::TimeUnit::compareTo (const TimeUnit & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

In the foregoing description, the notation sgn(expression) designates the mathematical signum function, which is defined to return one of -1, 0, or 1 according to whether the value of expression is negative, zero or positive. The implementor must ensure sgn(x.compareTo(y)) == -sgn(y.compareTo(x)) for all x and y. (This implies that x.compareTo(y) must throw an exception iff y.compareTo(x) throws an exception.)

The implementor must also ensure that the relation is transitive: (x.compareTo(y)>0 && y.compareTo(z)>0) implies x.compareTo(z)>0.

Finally, the implementer must ensure that x.compareTo(y)==0 implies that sgn(x.compareTo(z)) == sgn(y.compareTo(z)), for all z.

It is strongly recommended, but not strictly required that (x.compareTo(y)==0) == (x.equals(y)). Generally speaking, any class that implements the Comparable interface and violates this condition should clearly indicate this fact. The recommended language is "Note: this class has a natural ordering that is inconsistent with equals."

Parameters:

value - the Object to be compared.

Returns:

a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

6.551.3.2 long long decaf::util::concurrent::TimeUnit::convert (long long *sourceDuration*, const TimeUnit & *sourceUnit*) const

Convert the given time duration in the given unit to this unit. Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN_VALUE if negative or Long.MAX_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: TimeUnit.MILLISECONDS.convert(10L, **TimeUnit.MINUTES** (p. 2570))

Parameters:

sourceDuration - Duration value to convert.

sourceUnit - Unit type of the source duration.

Returns:

the converted duration in this unit, or Long.MIN_VALUE if conversion would negatively overflow, or Long.MAX_VALUE if it would positively overflow.

6.551.3.3 virtual bool decaf::util::concurrent::TimeUnit::equals (const TimeUnit & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.551.3.4 virtual bool decaf::util::concurrent::TimeUnit::operator< (const TimeUnit & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.551.3.5 virtual bool decaf::util::concurrent::TimeUnit::operator== (const TimeUnit & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.551.3.6 void decaf::util::concurrent::TimeUnit::sleep (long long *timeout*) const

Perform a timed `Thread.join` using this time unit. This is a convenience method that converts time arguments into the form required by the `Thread.join` method.

Parameters:

thread the thread to wait for

timeout the maximum time to wait

Exceptions:

InterruptedException if interrupted while waiting.

See also:

Thread::join(long, int) Perform a `Thread.sleep` using this unit. This is a convenience method that converts time arguments into the form required by the `Thread.sleep` method.

Parameters:

timeout the minimum time to sleep

See also:

Thread::sleep

6.551.3.7 void decaf::util::concurrent::TimeUnit::timedWait (Synchronizable * *obj*, long long *timeout*) const

Perform a timed `Object.wait` using this time unit. This is a convenience method that converts timeout arguments into the form required by the `Object.wait` method.

For example, you could implement a blocking `poll` method (see **BlockingQueue.poll** (p. ??)) using:

```
public synchronized  Object poll(long timeout, TimeUnit unit) throws InterruptedException {
  while (empty) {
    unit.timedWait(this, timeout);
    ...
  }
}
```

Parameters:

obj the object to wait on

timeout the maximum time to wait.

Exceptions:

InterruptedException if interrupted while waiting.

See also:

Synchronizable::wait(long, int)

6.551.3.8 long long decaf::util::concurrent::TimeUnit::toDays (long long *duration*) const [inline]

Equivalent to `DAYS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

convert (p. 2565)

6.551.3.9 long long decaf::util::concurrent::TimeUnit::toHours (long long *duration*) const [inline]

Equivalent to `HOURS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

convert (p. 2565)

6.551.3.10 long long decaf::util::concurrent::TimeUnit::toMicros (long long *duration*) const [inline]

Equivalent to `MICROSECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

convert (p. 2565)

6.551.3.11 long long decaf::util::concurrent::TimeUnit::toMillis (long long *duration*) const [inline]

Equivalent to `MILLISECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

convert (p. 2565)

6.551.3.12 long long decaf::util::concurrent::TimeUnit::toMinutes (long long *duration*) const [inline]

Equivalent to `MINUTES.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

convert (p. 2565)

6.551.3.13 long long decaf::util::concurrent::TimeUnit::toNanos (long long *duration*) const [inline]

Equivalent to `NANOSECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration, or `Long.MIN_VALUE` if conversion would negatively overflow, or `Long.MAX_VALUE` if it would positively overflow.

See also:

convert (p. 2565)

6.551.3.14 long long decaf::util::concurrent::TimeUnit::toSeconds (long long *duration*) const [inline]

Equivalent to `SECONDS.convert(duration, this)`.

Parameters:

duration the duration

Returns:

the converted duration.

See also:

convert (p. 2565)

6.551.3.15 virtual std::string decaf::util::concurrent::TimeUnit::toString () const [virtual]

Converts the **TimeUnit** (p. 2562) type to the Name of the **TimeUnit** (p. 2562).

Returns:

String name of the **TimeUnit** (p. 2562)

6.551.3.16 static const TimeUnit& decaf::util::concurrent::TimeUnit::valueOf (const std::string & *name*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Returns the **TimeUnit** (p. 2562) constant of this type with the specified name. The string must match exactly an identifier used to declare an **TimeUnit** (p. 2562) constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:

name The Name of the **TimeUnit** (p. 2562) constant to be returned.

Returns:

A constant reference to the **TimeUnit** (p. 2562) Constant with the given name.

Exceptions:

IllegalArgumentException if this enum type has no constant with the specified name

6.551.4 Field Documentation

6.551.4.1 const TimeUnit decaf::util::concurrent::TimeUnit::DAYS [static]

6.551.4.2 const TimeUnit decaf::util::concurrent::TimeUnit::HOURS [static]

6.551.4.3 const TimeUnit decaf::util::concurrent::TimeUnit::MICROSECONDS [static]

6.551.4.4 const TimeUnit decaf::util::concurrent::TimeUnit::MILLISECONDS [static]

6.551.4.5 const TimeUnit decaf::util::concurrent::TimeUnit::MINUTES [static]

6.551.4.6 const TimeUnit decaf::util::concurrent::TimeUnit::NANOSECONDS [static]

The Actual **TimeUnit** (p. 2562) enumerations.

6.551.4.7 const TimeUnit decaf::util::concurrent::TimeUnit::SECONDS [static]

6.551.4.8 const TimeUnit* const decaf::util::concurrent::TimeUnit::values[] [static]

The An Array of **TimeUnit** (p. 2562) Instances.

The documentation for this class was generated from the following file:

- src/main/decaf/util/concurrent/**TimeUnit.h**

6.552 cms::Topic Class Reference

An interface encapsulating a provider-specific topic name.

`#include <src/main/cms/Topic.h>`Inheritance diagram for cms::Topic:

Public Member Functions

- virtual ~**Topic** ()
- virtual std::string **getTopicName** () const =0 throw (CMSException)

  Gets the name of this topic.

6.552.1 Detailed Description

An interface encapsulating a provider-specific topic name. A **Topic** (p. 2571) is a Publish / Subscribe type **Destination** (p. 1190). All Messages sent to a **Topic** (p. 2571) are broadcast to all Subscribers of that **Topic** (p. 2571) unless the Subscriber defines a **Message** (p. 1753) selector that filters out that **Message** (p. 1753).

Since:

1.0

6.552.2 Constructor & Destructor Documentation

6.552.2.1 virtual cms::Topic::~Topic () `[inline, virtual]`

6.552.3 Member Function Documentation

6.552.3.1 virtual std::string cms::Topic::getTopicName () const throw (CMSException) `[pure virtual]`

Gets the name of this topic.

Returns:

The topic name.

Exceptions:

CMSException *(*p. ***850)*** - If an internal error occurs.

Implemented in **activemq::commands::ActiveMQTopic** (p. 468).

The documentation for this class was generated from the following file:

- src/main/cms/**Topic.h**

6.553 activemq::state::Tracked Class Reference

`#include <src/main/activemq/state/Tracked.h>`Inheritance diagram for activemq::state::Tracked:

Public Member Functions

- **Tracked** ()
- **Tracked** (const **Pointer**< **decaf::lang::Runnable** > &runnable)
- virtual ~**Tracked** ()
- void **onResponse** ()
- bool **isWaitingForResponse** () const

6.553.1 Constructor & Destructor Documentation

6.553.1.1 activemq::state::Tracked::Tracked () `[inline]`

6.553.1.2 activemq::state::Tracked::Tracked (const Pointer< decaf::lang::Runnable > & *runnable*)

6.553.1.3 virtual activemq::state::Tracked::~Tracked () `[inline, virtual]`

6.553.2 Member Function Documentation

6.553.2.1 bool activemq::state::Tracked::isWaitingForResponse () const `[inline]`

6.553.2.2 void activemq::state::Tracked::onResponse ()

The documentation for this class was generated from the following file:

- src/main/activemq/state/**Tracked.h**

6.554 activemq::commands::TransactionId Class Reference

`#include <src/main/activemq/commands/TransactionId.h>`Inheritance diagram for activemq::commands::TransactionId:

Public Types

- typedef **decaf::lang::PointerComparator< TransactionId > COMPARATOR**

Public Member Functions

- **TransactionId** ()
- **TransactionId** (const **TransactionId** &other)
- virtual ~**TransactionId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **TransactionId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual int **compareTo** (const **TransactionId** &value) const
- virtual bool **equals** (const **TransactionId** &value) const
- virtual bool **operator==** (const **TransactionId** &value) const
- virtual bool **operator<** (const **TransactionId** &value) const
- **TransactionId** & **operator=** (const **TransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONID** = 0

6.554.1 Member Typedef Documentation

6.554.1.1 typedef decaf::lang::PointerComparator<TransactionId> activemq::commands::TransactionId::COMPARATOR

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1601), and **activemq::commands::XATransactionId** (p. 2732).

6.554.2 Constructor & Destructor Documentation

6.554.2.1 activemq::commands::TransactionId::TransactionId ()

6.554.2.2 activemq::commands::TransactionId::TransactionId (const TransactionId & *other*)

6.554.2.3 virtual activemq::commands::TransactionId::~TransactionId () [virtual]

6.554.3 Member Function Documentation

6.554.3.1 virtual TransactionId* activemq::commands::TransactionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1601), and **activemq::commands::XATransactionId** (p. 2732).

6.554.3.2 virtual int activemq::commands::TransactionId::compareTo (const TransactionId & *value*) const [virtual]

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1601), and **activemq::commands::XATransactionId** (p. 2732).

6.554.3.3 virtual void activemq::commands::TransactionId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1601), and **activemq::commands::XATransactionId** (p. 2732).

6.554.3.4 virtual bool activemq::commands::TransactionId::equals (const TransactionId & *value*) const [virtual]

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1602), and **activemq::commands::XATransactionId** (p. 2733).

6.554.3.5 virtual bool activemq::commands::TransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1602), and **activemq::commands::XATransactionId** (p. 2733).

6.554.3.6 virtual unsigned char activemq::commands::TransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1602), and **activemq::commands::XATransactionId** (p. 2733).

6.554.3.7 virtual bool activemq::commands::TransactionId::operator< (const TransactionId & *value*) const [virtual]

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1602), and **activemq::commands::XATransactionId** (p. 2734).

6.554.3.8 TransactionId& activemq::commands::TransactionId::operator= (const TransactionId & *other*)

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1602), and **activemq::commands::XATransactionId** (p. 2734).

6.554.3.9 virtual bool activemq::commands::TransactionId::operator== (const TransactionId & *value*) const [virtual]

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1603), and **activemq::commands::XATransactionId** (p. 2734).

6.554.3.10 virtual std::string activemq::commands::TransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseDataStructure** (p. 560).

Reimplemented in **activemq::commands::LocalTransactionId** (p. 1603), and **activemq::commands::XATransactionId** (p. 2734).

6.554.4 Field Documentation

6.554.4.1 const unsigned char activemq::commands::TransactionId::ID_TRANSACTIONID = 0 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionId.h**

6.555 activemq::wireformat::openwire::marshal::v1::TransactionIdMarsh Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2577).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual ~**TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.555.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2577). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.555.2 Constructor & Destructor Documentation

6.555.2.1 activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::TransactionIdMars () [inline]

6.555.2.2 virtual activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::~TransactionIdMa () [inline, virtual]

6.555.3 Member Function Documentation

6.555.3.1 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1613), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 2737).

6.555.3.2 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1614), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 2738).

6.555.3.3 virtual int activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1614), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 2738).

6.555.3.4 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1615), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 2739).

6.555.3.5 virtual void activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller** (p. 1615), and **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller** (p. 2739).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionIdMarshaller.h**

6.556 activemq::wireformat::openwire::marshal::v3::TransactionIdMarsh Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2581).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual ~**TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.556.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2581). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.556.2 Constructor & Destructor Documentation

6.556.2.1 activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::TransactionIdMars () [inline]

6.556.2.2 virtual activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::~TransactionIdMa () [inline, virtual]

6.556.3 Member Function Documentation

6.556.3.1 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1609), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2741).

6.556.3.2 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1610), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2742).

6.556.3.3 virtual int activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1610), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2742).

6.556.3.4 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1611), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2743).

6.556.3.5 virtual void activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller** (p. 1611), and **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller** (p. 2743).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**TransactionIdMarshaller.h**

6.557 activemq::wireformat::openwire::marshal::v2::TransactionIdMarsh Class Reference

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2585).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller:

Public Member Functions

- **TransactionIdMarshaller** ()
- virtual ~**TransactionIdMarshaller** ()
- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.557.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionIdMarshaller** (p. 2585). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.557.2 Constructor & Destructor Documentation

6.557.2.1 activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::TransactionIdMars () [inline]

6.557.2.2 virtual activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::~TransactionIdMa () [inline, virtual]

6.557.3 Member Function Documentation

6.557.3.1 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1605), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2745).

6.557.3.2 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1606), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2746).

6.557.3.3 virtual int activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1606), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2746).

6.557.3.4 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1607), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2747).

6.557.3.5 virtual void activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

Reimplemented in **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller** (p. 1607), and **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller** (p. 2747).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionIdMarshaller.h**

6.558 activemq::commands::TransactionInfo Class Reference

`#include <src/main/activemq/commands/TransactionInfo.h>`Inheritance diagram for activemq::commands::TransactionInfo:

Public Member Functions

- **TransactionInfo** ()
- virtual ~**TransactionInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **TransactionInfo** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174) passed in to this one, and returns if they are equivalent.*

- virtual const **Pointer**< **ConnectionId** > & **getConnectionId** () const
- virtual **Pointer**< **ConnectionId** > & **getConnectionId** ()
- virtual void **setConnectionId** (const **Pointer**< **ConnectionId** > &**connectionId**)
- virtual const **Pointer**< **TransactionId** > & **getTransactionId** () const
- virtual **Pointer**< **TransactionId** > & **getTransactionId** ()
- virtual void **setTransactionId** (const **Pointer**< **TransactionId** > &**transactionId**)
- virtual unsigned char **getType** () const
- virtual void **setType** (unsigned char **type**)
- virtual bool **isTransactionInfo** () const
- virtual **Pointer**< **Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

Static Public Attributes

- static const unsigned char **ID_TRANSACTIONINFO** = 7

Protected Member Functions

- **TransactionInfo** (const **TransactionInfo** &)
- **TransactionInfo** & **operator**= (const **TransactionInfo** &)

Protected Attributes

- **Pointer< ConnectionId > connectionId**
- **Pointer< TransactionId > transactionId**
- unsigned char **type**

6.558.1 Constructor & Destructor Documentation

6.558.1.1 activemq::commands::TransactionInfo::TransactionInfo (const TransactionInfo &) [inline, protected]

6.558.1.2 activemq::commands::TransactionInfo::TransactionInfo ()

6.558.1.3 virtual activemq::commands::TransactionInfo::~TransactionInfo () [virtual]

6.558.2 Member Function Documentation

6.558.2.1 virtual TransactionInfo* activemq::commands::TransactionInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.558.2.2 virtual void activemq::commands::TransactionInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.558.2.3 virtual bool activemq::commands::TransactionInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.558.2.4 virtual Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () [virtual]

6.558.2.5 virtual const Pointer<ConnectionId>& activemq::commands::TransactionInfo::getConnectionId () const [virtual]

6.558.2.6 virtual unsigned char activemq::commands::TransactionInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.558.2.7 virtual Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () [virtual]

6.558.2.8 virtual const Pointer<TransactionId>& activemq::commands::TransactionInfo::getTransactionId () const [virtual]

6.558.2.9 virtual unsigned char activemq::commands::TransactionInfo::getType () const [virtual]

6.558.2.10 virtual bool activemq::commands::TransactionInfo::isTransactionInfo () const [inline, virtual]

Returns:

an answer of true to the **isTransactionInfo()** (p. 2591) query.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.558.2.11 TransactionInfo& activemq::commands::TransactionInfo::operator= (const TransactionInfo &) [inline, protected]

6.558.2.12 virtual void activemq::commands::TransactionInfo::setConnectionId (const Pointer< ConnectionId > & *connectionId*) [virtual]

6.558.2.13 virtual void activemq::commands::TransactionInfo::setTransactionId (const Pointer< TransactionId > & *transactionId*) [virtual]

6.558.2.14 virtual void activemq::commands::TransactionInfo::setType (unsigned char *type*) [virtual]

6.558.2.15 virtual std::string activemq::commands::TransactionInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.558.2.16 virtual Pointer<Command> activemq::commands::TransactionInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.558.3 Field Documentation

6.558.3.1 Pointer<ConnectionId> activemq::commands::TransactionInfo::connectionId [protected]

6.558.3.2 const unsigned char activemq::commands::TransactionInfo::ID_TRANSACTIONINFO = 7 [static]

6.558.3.3 Pointer<TransactionId> activemq::commands::TransactionInfo::transactionId [protected]

6.558.3.4 unsigned char activemq::commands::TransactionInfo::type [protected]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**TransactionInfo.h**

6.559 activemq::wireformat::openwire::marshal::v2::TransactionInfoMars Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2594).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual ~**TransactionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.559.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2594). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.559.2 Constructor & Destructor Documentation

6.559.2.1 activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::TransactionInfoM () [inline]

6.559.2.2 virtual activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::~TransactionInfo () [inline, virtual]

6.559.3 Member Function Documentation

6.559.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.559.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.559.3.3 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 529).

6.559.3.4 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 530).

6.559.3.5 virtual int activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 531).

6.559.3.6 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 532).

6.559.3.7 virtual void activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller** (p. 533).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**TransactionInfoMarshaller.h**

6.560 activemq::wireformat::openwire::marshal::v3::TransactionInfoMars Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2598).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual ~**TransactionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.560.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2598). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.560.2 Constructor & Destructor Documentation

6.560.2.1 activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::TransactionInfoM () [inline]

6.560.2.2 virtual activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::~TransactionInfo () [inline, virtual]

6.560.3 Member Function Documentation

6.560.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.560.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.560.3.3 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 515).

6.560.3.4 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 516).

6.560.3.5 virtual int activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 517).

6.560.3.6 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 518).

6.560.3.7 virtual void activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller** (p. 519).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**TransactionInfoMarshaller.h**

6.561 activemq::wireformat::openwire::marshal::v1::TransactionInfoMars Class Reference

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2602).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller:

Public Member Functions

- **TransactionInfoMarshaller** ()
- virtual ~**TransactionInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.561.1 Detailed Description

Marshaling code for Open Wire Format for **TransactionInfoMarshaller** (p. 2602). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.561.2 Constructor & Destructor Documentation

6.561.2.1 activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::TransactionInfoM () [inline]

6.561.2.2 virtual activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::~TransactionInfo () [inline, virtual]

6.561.3 Member Function Documentation

6.561.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.561.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.561.3.3 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 522).

6.561.3.4 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 523).

6.561.3.5 virtual int activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 524).

6.561.3.6 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 525).

6.561.3.7 virtual void activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller** (p. 526).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**TransactionInfoMarshaller.h**

6.562 activemq::state::TransactionState Class Reference

`#include <src/main/activemq/state/TransactionState.h>`

Public Member Functions

- **TransactionState** (const **Pointer**< **TransactionId** > &id)
- virtual ~**TransactionState** ()
- std::string **toString** () const
- void **addCommand** (const **Pointer**< **Command** > &operation)
- void **checkShutdown** () const
- void **shutdown** ()
- const **StlList**< **Pointer**< **Command** > > & **getCommands** () const
- const **Pointer**< **TransactionId** > & **getId** () const
- void **setPrepared** (bool prepared)
- bool **isPrepared** () const
- void **setPreparedResult** (int preparedResult)
- int **getPreparedResult** () const

6.562.1 Constructor & Destructor Documentation

6.562.1.1 activemq::state::TransactionState::TransactionState (const Pointer< TransactionId > & *id*)

6.562.1.2 virtual activemq::state::TransactionState::~TransactionState () [virtual]

6.562.2 Member Function Documentation

6.562.2.1 void activemq::state::TransactionState::addCommand (const Pointer< Command > & *operation*)

6.562.2.2 void activemq::state::TransactionState::checkShutdown () const

6.562.2.3 const StlList< Pointer<Command> >& activemq::state::TransactionState::getCommands () const [inline]

6.562.2.4 const Pointer<TransactionId>& activemq::state::TransactionState::getId () const [inline]

6.562.2.5 int activemq::state::TransactionState::getPreparedResult () const [inline]

6.562.2.6 bool activemq::state::TransactionState::isPrepared () const [inline]

6.562.2.7 void activemq::state::TransactionState::setPrepared (bool *prepared*) [inline]

6.562.2.8 void activemq::state::TransactionState::setPreparedResult (int *preparedResult*) [inline]

6.562.2.9 void activemq::state::TransactionState::shutdown () [inline]

6.562.2.10 std::string activemq::state::TransactionState::toString () const

The documentation for this class was generated from the following file:

- src/main/activemq/state/**TransactionState.h**

6.563 activemq::transport::Transport Class Reference

Interface for a **transport** (p. 67) layer for command objects.

`#include <src/main/activemq/transport/Transport.h>`Inheritance diagram for activemq::transport::Transport:

Public Member Functions

- virtual ~**Transport** ()
- virtual void **oneway** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends a one-way command.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout)=0 throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)

  Sends the given command to the broker and then waits for the response.

- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)=0

  Sets the WireFormat instance to use.

- virtual void **setTransportListener** (**TransportListener** *listener)=0

  *Sets the observer of asynchronous events from this **transport** (p. 67).*

- virtual **TransportListener** * **getTransportListener** () const =0

  *Gets the observer of asynchronous events from this **transport** (p. 67).*

- virtual **Transport** * **narrow** (const std::type_info &typeId)=0

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const =0

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const =0

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const =0

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const =0
- virtual void **reconnect** (const **decaf::net::URI** &uri)=0 throw (decaf::io::IOException)

  reconnect to another location

6.563.1 Detailed Description

Interface for a **transport** (p. 67) layer for command objects. Callers can send oneway messages or make synchronous requests. Non-response messages will be delivered to the specified listener object upon receipt. A user of the **Transport** (p. 2608) can set an exception listener to be notified of errors that occurs in Threads that the **Transport** (p. 2608) layer runs. Transports should be given an instance of a WireFormat object when created so that they can turn the built in Commands to / from the required wire format encoding.

6.563.2 Constructor & Destructor Documentation

6.563.2.1 virtual activemq::transport::Transport::~Transport () [inline, virtual]

6.563.3 Member Function Documentation

6.563.3.1 virtual std::string activemq::transport::Transport::getRemoteAddress () const [pure virtual]

Returns:

the remote address for this connection

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1300), **activemq::transport::IOTransport** (p. 1482), **activemq::transport::mock::MockTransport** (p. 1913), and **activemq::transport::TransportFilter** (p. 2619).

6.563.3.2 virtual TransportListener* activemq::transport::Transport::getTransportListener () const [pure virtual]

Gets the observer of asynchronous events from this **transport** (p. 67).

Returns:

the listener of **transport** (p. 67) events.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1300), **activemq::transport::IOTransport** (p. 1482), **activemq::transport::mock::MockTransport** (p. 1913), and **activemq::transport::TransportFilter** (p. 2619).

6.563.3.3 virtual bool activemq::transport::Transport::isClosed () const [pure virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1301), **activemq::transport::IOTransport** (p. 1482), **activemq::transport::mock::MockTransport** (p. 1913), **activemq::transport::tcp::TcpTransport** (p. 2533), and **activemq::transport::TransportFilter** (p. 2619).

6.563.3.4 virtual bool activemq::transport::Transport::isConnected () const [pure virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1301), **activemq::transport::IOTransport** (p. 1483), **activemq::transport::mock::MockTransport** (p. 1914), **activemq::transport::tcp::TcpTransport** (p. 2533), and **activemq::transport::TransportFilter** (p. 2619).

6.563.3.5 virtual bool activemq::transport::Transport::isFaultTolerant () const [pure virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1301), **activemq::transport::IOTransport** (p. 1483), **activemq::transport::mock::MockTransport** (p. 1914), **activemq::transport::tcp::TcpTransport** (p. 2534), and **activemq::transport::TransportFilter** (p. 2619).

6.563.3.6 virtual Transport* activemq::transport::Transport::narrow (const std::type_info & *typeId*) [pure virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1302), **activemq::transport::IOTransport** (p. 1483), **activemq::transport::mock::MockTransport** (p. 1914), and **activemq::transport::TransportFilter** (p. 2620).

Referenced by activemq::transport::TransportFilter::narrow(), and activemq::transport::failover::FailoverTransport::narrow().

6.563.3.7 virtual void activemq::transport::Transport::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::failover::FailoverTransport** (p. 1303), **activemq::transport::IOTransport** (p. 1483), **activemq::transport::logging::LoggingTransport** (p. 1638), **activemq::transport::mock::MockTransport** (p. 1914), **activemq::transport::TransportFilter** (p. 2620), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1985).

6.563.3.8 virtual void activemq::transport::Transport::reconnect (const decaf::net::URI & *uri*) throw (decaf::io::IOException) [pure virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1303), and **activemq::transport::TransportFilter** (p. 2621).

6.563.3.9 virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command - The command to be sent.

timeout - The time to wait for this response.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2239), **activemq::transport::failover::FailoverTransport** (p. 1304), **activemq::transport::IOTransport** (p. 1484), **activemq::transport::logging::LoggingTransport** (p. 1638), **activemq::transport::mock::MockTransport** (p. 1915), **activemq::transport::TransportFilter** (p. 2621), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1986).

6.563.3.10 virtual Pointer<Response> activemq::transport::Transport::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

Sends the given command to the broker and then waits for the response.

Parameters:

command the command to be sent.

Returns:

the response from the broker.

Exceptions:

IOException if an exception occurs during the read of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2239), **activemq::transport::failover::FailoverTransport** (p. 1304), **activemq::transport::IOTransport** (p. 1484), **activemq::transport::logging::LoggingTransport** (p. 1639), **activemq::transport::mock::MockTransport** (p. 1915), **activemq::transport::TransportFilter** (p. 2622), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1986).

6.563.3.11 virtual void activemq::transport::Transport::setTransportListener (TransportListener * *listener*) [pure virtual]

Sets the observer of asynchronous events from this **transport** (p. 67).

Parameters:

listener the listener of **transport** (p. 67) events.

Implemented in **activemq::transport::failover::FailoverTransport** (p. 1306), **activemq::transport::IOTransport** (p. 1485), **activemq::transport::mock::MockTransport** (p. 1917), and **activemq::transport::TransportFilter** (p. 2622).

6.563.3.12 virtual void activemq::transport::Transport::setWireFormat (const Pointer< wireformat::WireFormat > & *wireFormat*) [pure virtual]

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode **commands** (p. 59).

Implemented in **activemq::transport::IOTransport** (p. 1485), and **activemq::transport::TransportFilter** (p. 2622).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**Transport.h**

6.564 activemq::transport::TransportFactory Class Reference

Defines the interface for Factories that create Transports or TransportFilters.

`#include <src/main/activemq/transport/TransportFactory.h>`Inheritance diagram for activemq::transport::TransportFactory:

Public Member Functions

- virtual ~**TransportFactory** ()
- virtual **Pointer< Transport > create** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)

  *Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.*

- virtual **Pointer< Transport > createComposite** (const **decaf::net::URI** &location)=0 throw (exceptions::ActiveMQException)

  *Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.*

6.564.1 Detailed Description

Defines the interface for Factories that create Transports or TransportFilters. The factory should be able to create either a completely configured **Transport** (p. 2608) meaning that it has all the appropriate filters wrapping it, or it should be able to create a slimed down version that is used in composite transports like Failover or Fanout.

Since:

3.0

6.564.2 Constructor & Destructor Documentation

6.564.2.1 virtual activemq::transport::TransportFactory::~TransportFactory () [inline, virtual]

6.564.3 Member Function Documentation

6.564.3.1 virtual Pointer<Transport> activemq::transport::TransportFactory::create (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [pure virtual]

Creates a fully configured **Transport** (p. 2608) instance which could be a chain of filters and transports.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1309), **activemq::transport::mock::MockTransportFactory** (p. 1919), and **activemq::transport::tcp::TcpTransportFactory** (p. 2536).

6.564.3.2 virtual Pointer<Transport> activemq::transport::TransportFactory::createComposite (const decaf::net::URI & *location*) throw (exceptions::ActiveMQException) [pure virtual]

Creates a slimed down **Transport** (p. 2608) instance which can be used in composite **transport** (p. 67) instances.

Parameters:

location - URI location to connect to plus any properties to assign.

Exceptions:

ActiveMQexception if an error occurs

Implemented in **activemq::transport::failover::FailoverTransportFactory** (p. 1309), **activemq::transport::mock::MockTransportFactory** (p. 1919), and **activemq::transport::tcp::TcpTransportFactory** (p. 2536).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFactory.h**

6.565 activemq::transport::TransportFilter Class Reference

A filter on the **transport** (p. 67) layer.

`#include <src/main/activemq/transport/TransportFilter.h>`Inheritance diagram for activemq::transport::TransportFilter:

Public Member Functions

- **TransportFilter** (const **Pointer**< **Transport** > &**next**)
  Constructor.
- virtual ~**TransportFilter** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)
  Event handler for the receipt of a command.
- virtual void **onException** (const **decaf::lang::Exception** &ex)
  *Event handler for an exception from a command **transport** (p. 67).*
- virtual void **transportInterrupted** ()
  *The **transport** (p. 67) has suffered an interruption from which it hopes to recover.*
- virtual void **transportResumed** ()
  *The **transport** (p. 67) has resumed after an interruption.*
- virtual void **oneway** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
  Sends a one-way command.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
  Not supported by this class - throws an exception.
- virtual **Pointer**< **Response** > **request** (const **Pointer**< **Command** > &command, unsigned int timeout) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException)
  Not supported by this class - throws an exception.
- virtual void **setTransportListener** (**TransportListener** *****listener**)
  *Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*
- virtual **TransportListener** * **getTransportListener** () const
  *Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).*
- virtual void **setWireFormat** (const **Pointer**< **wireformat::WireFormat** > &wireFormat)
  Sets the WireFormat instance to use.
- virtual void **start** () throw (cms::CMSException)

*Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59).*

- virtual void **close** () throw (cms::CMSException)

  Stops the polling thread and closes the streams.

- virtual **Transport** * **narrow** (const std::type_info &typeId)

  *Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.*

- virtual bool **isFaultTolerant** () const

  *Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.*

- virtual bool **isConnected** () const

  *Is the **Transport** (p. 2608) Connected to its Broker.*

- virtual bool **isClosed** () const

  *Has the **Transport** (p. 2608) been shutdown and no longer usable.*

- virtual std::string **getRemoteAddress** () const
- virtual void **reconnect** (const **decaf::net::URI** &uri) throw (decaf::io::IOException)

  reconnect to another location

Protected Member Functions

- void **fire** (const **decaf::lang::Exception** &ex)

  Notify the exception listener.

- void **fire** (const **Pointer< Command >** &command)

  Notify the command listener.

Protected Attributes

- **Pointer< Transport > next**

  *The **transport** (p. 67) that this filter wraps around.*

- **TransportListener** * **listener**

  *Listener of this **transport** (p. 67).*

6.565.1 Detailed Description

A filter on the **transport** (p. 67) layer. **Transport** (p. 2608) filters implement the **Transport** (p. 2608) interface and optionally delegate calls to another **Transport** (p. 2608) object.

6.565.2 Constructor & Destructor Documentation

6.565.2.1 activemq::transport::TransportFilter::TransportFilter (const Pointer< Transport > & *next*)

Constructor.

Parameters:

next - the next **Transport** (p. 2608) in the chain

6.565.2.2 virtual activemq::transport::TransportFilter::~TransportFilter () [inline, virtual]

6.565.3 Member Function Documentation

6.565.3.1 virtual void activemq::transport::TransportFilter::close () throw (cms::CMSException) [inline, virtual]

Stops the polling thread and closes the streams. This can be called explicitly, but is also called in the destructor. Once this object has been closed, it cannot be restarted.

Exceptions:

CMSException if errors occur.

Implements **cms::Closeable** (p. 838).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::tcp::TcpTransport** (p. 2533), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1985).

6.565.3.2 void activemq::transport::TransportFilter::fire (const Pointer< Command > & *command*) [inline, protected]

Notify the command listener.

Parameters:

command - the command to send to the listener

6.565.3.3 void activemq::transport::TransportFilter::fire (const decaf::lang::Exception & *ex*) [inline, protected]

Notify the exception listener.

Parameters:

ex - the exception to send to listeners

6.565.3.4 virtual std::string activemq::transport::TransportFilter::getRemoteAddress () const [inline, virtual]

Returns:

the remote address for this connection

Implements **activemq::transport::Transport** (p. 2609).

6.565.3.5 virtual TransportListener* activemq::transport::TransportFilter::getTransportListener () const [inline, virtual]

Gets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Returns:

The listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2609).

6.565.3.6 virtual bool activemq::transport::TransportFilter::isClosed () const [inline, virtual]

Has the **Transport** (p. 2608) been shutdown and no longer usable.

Returns:

true if the **Transport** (p. 2608)

Implements **activemq::transport::Transport** (p. 2609).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2533).

6.565.3.7 virtual bool activemq::transport::TransportFilter::isConnected () const [inline, virtual]

Is the **Transport** (p. 2608) Connected to its Broker.

Returns:

true if a connection has been made.

Implements **activemq::transport::Transport** (p. 2610).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2533).

6.565.3.8 virtual bool activemq::transport::TransportFilter::isFaultTolerant () const [inline, virtual]

Is this **Transport** (p. 2608) fault tolerant, meaning that it will reconnect to a broker on disconnect.

Returns:

true if the **Transport** (p. 2608) is fault tolerant.

Implements **activemq::transport::Transport** (p. 2610).

Reimplemented in **activemq::transport::tcp::TcpTransport** (p. 2534).

6.565.3.9 virtual Transport* activemq::transport::TransportFilter::narrow (const std::type_info & *typeId*) [inline, virtual]

Narrows down a Chain of Transports to a specific **Transport** (p. 2608) to allow a higher level **transport** (p. 67) to skip intermediate Transports in certain circumstances.

Parameters:

typeId - The type_info of the Object we are searching for.

Returns:

the requested Object. or NULL if its not in this chain.

Implements **activemq::transport::Transport** (p. 2610).

References activemq::transport::Transport::narrow().

6.565.3.10 virtual void activemq::transport::TransportFilter::onCommand (const Pointer< Command > & *command*) [inline, virtual]

Event handler for the receipt of a command.

Parameters:

command - the received command object.

Implements **activemq::transport::TransportListener** (p. 2624).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::logging::LoggingTransport** (p. 1638), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1985).

6.565.3.11 virtual void activemq::transport::TransportFilter::oneway (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Sends a one-way command. Does not wait for any response from the broker.

Parameters:

command the command to be sent.

Exceptions:

IOException if an exception occurs during writing of the command.

UnsupportedOperationException if this method is not implemented by this **transport** (p. 67).

Implements **activemq::transport::Transport** (p. 2610).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::logging::LoggingTransport** (p. 1638), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1985).

6.565.3.12 virtual void activemq::transport::TransportFilter::onException (const decaf::lang::Exception & *ex*) [virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

ex The exception to handle.

Implements **activemq::transport::TransportListener** (p. 2625).

6.565.3.13 virtual void activemq::transport::TransportFilter::reconnect (const decaf::net::URI & *uri*) throw (decaf::io::IOException) [virtual]

reconnect to another location

Parameters:

uri

Exceptions:

IOException on failure of if not supported

Implements **activemq::transport::Transport** (p. 2611).

6.565.3.14 virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command*, unsigned int *timeout*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Not supported by this class - throws an exception.

Parameters:

command - The command that is sent as a request

timeout - The the time to wait for a response.

Exceptions:

IOException

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2611).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2239), **activemq::transport::logging::LoggingTransport** (p. 1638), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1986).

6.565.3.15 virtual Pointer<Response> activemq::transport::TransportFilter::request (const Pointer< Command > & *command*) throw (decaf::io::IOException, decaf::lang::exceptions::UnsupportedOperationException) [inline, virtual]

Not supported by this class - throws an exception.

Parameters:

command the command that is sent as a request

Exceptions:

IOException

UnsupportedOperationException.

Implements **activemq::transport::Transport** (p. 2612).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2239), **activemq::transport::logging::LoggingTransport** (p. 1639), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1986).

6.565.3.16 virtual void activemq::transport::TransportFilter::setTransportListener (TransportListener * *listener*) [inline, virtual]

Sets the observer of asynchronous **exceptions** (p. 62) from this **transport** (p. 67).

Parameters:

listener the listener of **transport** (p. 67) events.

Implements **activemq::transport::Transport** (p. 2612).

6.565.3.17 virtual void activemq::transport::TransportFilter::setWireFormat (const Pointer< wireformat::WireFormat > & *wireFormat*) [inline, virtual]

Sets the WireFormat instance to use.

Parameters:

wireFormat The WireFormat the object used to encode / decode **commands** (p. 59).

Implements **activemq::transport::Transport** (p. 2612).

6.565.3.18 virtual void activemq::transport::TransportFilter::start () throw (cms::CMSException) [inline, virtual]

Starts this **transport** (p. 67) object and creates the thread for polling on the input stream for **commands** (p. 59). If this object has been closed, throws an exception. Before calling start, the caller must set the IO streams and the reader and writer objects.

Exceptions:

CMSException if an error occurs or if this **transport** (p. 67) has already been closed.

Implements **cms::Startable** (p. 2413).

Reimplemented in **activemq::transport::correlator::ResponseCorrelator** (p. 2240), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1987).

6.565.3.19 virtual void activemq::transport::TransportFilter::transportInterrupted () [inline, virtual]

The **transport** (p. 67) has suffered an interruption from which it hopes to recover.

Implements **activemq::transport::TransportListener** (p. 2625).

6.565.3.20 virtual void activemq::transport::TransportFilter::transportResumed () [inline, virtual]

The **transport** (p. 67) has resumed after an interruption.

Implements **activemq::transport::TransportListener** (p. 2625).

6.565.4 Field Documentation

6.565.4.1 TransportListener* activemq::transport::TransportFilter::listener [protected]

Listener of this **transport** (p. 67).

6.565.4.2 Pointer<Transport> activemq::transport::TransportFilter::next [protected]

The **transport** (p. 67) that this filter wraps around.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportFilter.h**

6.566 activemq::transport::TransportListener Class Reference

A listener of asynchronous **exceptions** (p. 62) from a command **transport** (p. 67) object.

`#include <src/main/activemq/transport/TransportListener.h>`Inheritance diagram for activemq::transport::TransportListener:

Public Member Functions

- virtual ~**TransportListener** ()
- virtual void **onCommand** (const **Pointer**< **Command** > &command)=0

  Event handler for the receipt of a command.

- virtual void **onException** (const **decaf::lang::Exception** &ex)=0

  *Event handler for an exception from a command **transport** (p. 67).*

- virtual void **transportInterrupted** ()=0

  *The **transport** (p. 67) has suffered an interruption from which it hopes to recover.*

- virtual void **transportResumed** ()=0

  *The **transport** (p. 67) has resumed after an interruption.*

6.566.1 Detailed Description

A listener of asynchronous **exceptions** (p. 62) from a command **transport** (p. 67) object.

6.566.2 Constructor & Destructor Documentation

6.566.2.1 virtual activemq::transport::TransportListener::~TransportListener () [inline, virtual]

6.566.3 Member Function Documentation

6.566.3.1 virtual void activemq::transport::TransportListener::onCommand (const Pointer< Command > & *command*) [pure virtual]

Event handler for the receipt of a command. The **transport** (p. 67) passes off all received **commands** (p. 59) to its listeners, the listener then owns the Object. If there is no registered listener the **Transport** (p. 2608) deletes the command upon receipt.

Parameters:

command the received command object.

Implemented in **activemq::core::ActiveMQConnection** (p. 197), **activemq::transport::correlator::ResponseCorrelator** (p. 2238), **activemq::transport::failover::FailoverTransportListener** (p. 1312),

activemq::transport::logging::LoggingTransport (p. 1638), **activemq::transport::mock::InternalCommandListener** (p. 1457), **activemq::transport::TransportFilter** (p. 2620), and **activemq::wireformat::openwire::OpenWireFormatNegotiator** (p. 1985).

6.566.3.2 virtual void activemq::transport::TransportListener::onException (const decaf::lang::Exception & *ex*) [pure virtual]

Event handler for an exception from a command **transport** (p. 67).

Parameters:

ex The exception being propagated to this listener to handle.

Implemented in **activemq::core::ActiveMQConnection** (p. 197), **activemq::transport::failover::BackupTransport** (p. 502), **activemq::transport::failover::FailoverTransportListener** (p. 1312), and **activemq::transport::TransportFilter** (p. 2621).

6.566.3.3 virtual void activemq::transport::TransportListener::transportInterrupted () [pure virtual]

The **transport** (p. 67) has suffered an interruption from which it hopes to recover.

Implemented in **activemq::core::ActiveMQConnection** (p. 199), **activemq::transport::DefaultTransportListener** (p. 1186), **activemq::transport::failover::FailoverTransportListener** (p. 1312), and **activemq::transport::TransportFilter** (p. 2623).

6.566.3.4 virtual void activemq::transport::TransportListener::transportResumed () [pure virtual]

The **transport** (p. 67) has resumed after an interruption.

Implemented in **activemq::core::ActiveMQConnectionSupport** (p. 215), **activemq::transport::DefaultTransportListener** (p. 1186), **activemq::transport::failover::FailoverTransportListener** (p. 1312), and **activemq::transport::TransportFilter** (p. 2623).

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportListener.h**

6.567 activemq::transport::TransportRegistry Class Reference

Registry of all **Transport** (p. 2608) Factories that are available to the client at runtime.

`#include <src/main/activemq/transport/TransportRegistry.h>`

Public Member Functions

- virtual ~**TransportRegistry** ()
- **TransportFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)

  *Gets a Registered **TransportFactory** (p. 2614) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **TransportFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

  *Registers a new **TransportFactory** (p. 2614) with this Registry.*

- void **unregisterFactory** (const std::string &name)

  Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getTransportNames** () const

  Retrieves a list of the names of all the Registered Transport's in this Registry.

Static Public Member Functions

- static **TransportRegistry** & **getInstance** ()

  *Gets the single instance of the **TransportRegistry** (p. 2626).*

6.567.1 Detailed Description

Registry of all **Transport** (p. 2608) Factories that are available to the client at runtime. New Transport's must have a factory registered here before a connection attempt is made.

Since:

3.0

6.567.2 Constructor & Destructor Documentation

6.567.2.1 virtual activemq::transport::TransportRegistry::~TransportRegistry () [virtual]

6.567.3 Member Function Documentation

6.567.3.1 TransportFactory* activemq::transport::TransportRegistry::findFactory (const std::string & *name*) const throw (decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **TransportFactory** (p. 2614) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters:

name The name of the Factory to find in the Registry.

Returns:

the Factory registered under the given name.

Exceptions:

NoSuchElementException if no factory is registered with that name.

6.567.3.2 static TransportRegistry& activemq::transport::TransportRegistry::getInstance () [static]

Gets the single instance of the **TransportRegistry** (p. 2626).

Returns:

reference to the single instance of this Registry

6.567.3.3 std::vector<std::string> activemq::transport::TransportRegistry::getTransportNames () const

Retrieves a list of the names of all the Registered Transport's in this Registry.

Returns:

stl vector of strings with all the **Transport** (p. 2608) names registered.

6.567.3.4 void activemq::transport::TransportRegistry::registerFactory (const std::string & *name*, TransportFactory * *factory*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Registers a new **TransportFactory** (p. 2614) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters:

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions:

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.567.3.5 void activemq::transport::TransportRegistry::unregisterFactory (const std::string & *name*)

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters:

name Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/transport/**TransportRegistry.h**

6.568 decaf::net::UnknownHostException Class Reference

`#include <src/main/decaf/net/UnknownHostException.h>`Inheritance diagram for decaf::net::UnknownHostException:

Public Member Functions

- **UnknownHostException** () throw ()
  Default Constructor.
- **UnknownHostException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **UnknownHostException** (const **UnknownHostException** &ex) throw ()
  Copy Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **UnknownHostException** (const std::exception ***cause**) throw ()
  Constructor.
- **UnknownHostException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownHostException** * **clone** () const
  Clones this exception.
- virtual ~**UnknownHostException** () throw ()

6.568.1 Constructor & Destructor Documentation

6.568.1.1 decaf::net::UnknownHostException::UnknownHostException () throw () [inline]

Default Constructor.

6.568.1.2 decaf::net::UnknownHostException::UnknownHostException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.568.1.3 decaf::net::UnknownHostException::UnknownHostException (const UnknownHostException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.568.1.4 decaf::net::UnknownHostException::UnknownHostException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.568.1.5 decaf::net::UnknownHostException::UnknownHostException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.568.1.6 decaf::net::UnknownHostException::UnknownHostException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.568.1.7 virtual decaf::net::UnknownHostException::~UnknownHostException () throw () [inline, virtual]

6.568.2 Member Function Documentation

6.568.2.1 virtual UnknownHostException* decaf::net::UnknownHostException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownHostException.h**

6.569 decaf::net::UnknownServiceException Class Reference

`#include <src/main/decaf/net/UnknownServiceException.h>`Inheritance diagram for decaf::net::UnknownServiceException:

Public Member Functions

- **UnknownServiceException** () throw ()
  Default Constructor.
- **UnknownServiceException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **UnknownServiceException** (const **UnknownServiceException** &ex) throw ()
  Copy Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **UnknownServiceException** (const std::exception ***cause**) throw ()
  Constructor.
- **UnknownServiceException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnknownServiceException** * **clone** () const
  Clones this exception.
- virtual ~**UnknownServiceException** () throw ()

6.569.1 Constructor & Destructor Documentation

6.569.1.1 decaf::net::UnknownServiceException::UnknownServiceException () throw () [inline]

Default Constructor.

6.569.1.2 decaf::net::UnknownServiceException::UnknownServiceException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.569.1.3 decaf::net::UnknownServiceException::UnknownServiceException (const UnknownServiceException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

References decaf::lang::Exception::Exception().

6.569.1.4 decaf::net::UnknownServiceException::UnknownServiceException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.569.1.5 decaf::net::UnknownServiceException::UnknownServiceException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.569.1.6 decaf::net::UnknownServiceException::UnknownServiceException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.569.1.7 virtual decaf::net::UnknownServiceException::~UnknownServiceException () throw () [inline, virtual]

6.569.2 Member Function Documentation

6.569.2.1 virtual UnknownServiceException* decaf::net::UnknownServiceException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/net/**UnknownServiceException.h**

6.570 decaf::lang::exceptions::UnsupportedOperationException Class Reference

`#include <src/main/decaf/lang/exceptions/UnsupportedOperationException.h>`Inheritance diagram for decaf::lang::exceptions::UnsupportedOperationException:

Public Member Functions

- **UnsupportedOperationException** () throw ()
  Default Constructor.
- **UnsupportedOperationException** (const **Exception** &ex) throw ()
  Conversion Constructor from some other ***Exception*** *(*p. *1268**).*
- **UnsupportedOperationException** (const **UnsupportedOperationException** &ex) throw ()
  Copy Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **UnsupportedOperationException** (const std::exception ***cause**) throw ()
  Constructor.
- **UnsupportedOperationException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **UnsupportedOperationException** * **clone** () const
  Clones this exception.
- virtual ~**UnsupportedOperationException** () throw ()

6.570.1 Constructor & Destructor Documentation

6.570.1.1 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio () throw () [inline]

Default Constructor.

6.570.1.2 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other **Exception** (p. 1268).

Parameters:

ex An exception that should become this type of **Exception** (p. 1268)

6.570.1.3 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio (const UnsupportedOperationException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of **Exception** (p. 1268)

6.570.1.4 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.570.1.5 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause **Pointer** (p. 2011) to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.570.1.6 decaf::lang::exceptions::UnsupportedOperationException::UnsupportedOperationExceptio (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.570.1.7 virtual decaf::lang::exceptions::UnsupportedOperationException::~UnsupportedOperationExcept () throw () [inline, virtual]

6.570.2 Member Function Documentation

6.570.2.1 virtual UnsupportedOperationException* decaf::lang::exceptions::UnsupportedOperationException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

an new **Exception** (p. 1268) instance that is a copy of this one.

Reimplemented from **decaf::lang::Exception** (p. 1271).

Reimplemented in **decaf::nio::ReadOnlyBufferException** (p. 2169).

The documentation for this class was generated from the following file:

- src/main/decaf/lang/exceptions/**UnsupportedOperationException.h**

6.571 decaf::net::URI Class Reference

This class represents an instance of a **URI** (p. 2638) as defined by RFC 2396.

`#include <src/main/decaf/net/URI.h>`Inheritance diagram for decaf::net::URI:

Public Member Functions

- **URI** ()

  Default Constructor, same as calling a Constructor with all fields empty.

- **URI** (const **URI** &uri) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) as a copy of another **URI** (p. 2638).*

- **URI** (const std::string &uri) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) from the given string.*

- **URI** (const std::string &scheme, const std::string &ssp, const std::string &fragment) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) from the given components.*

- **URI** (const std::string &scheme, const std::string &userInfo, const std::string &host, int port, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) from the given components.*

- **URI** (const std::string &scheme, const std::string &host, const std::string &path, const std::string &fragment) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) from the given components.*

- **URI** (const std::string &scheme, const std::string &authority, const std::string &path, const std::string &query, const std::string &fragment) throw (URISyntaxException)

  *Constructs a **URI** (p. 2638) from the given components.*

- virtual ~**URI** ()
- virtual int **compareTo** (const **URI** &value) const

  Compares this object with the specified object for order.

- virtual bool **equals** (const **URI** &value) const
- virtual bool **operator==** (const **URI** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **URI** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- std::string **getAuthority** () const
- std::string **getFragment** () const
- std::string **getHost** () const

- std::string **getPath** () const
- int **getPort** () const
- std::string **getQuery** () const
- std::string **getScheme** () const
- std::string **getUserInfo** () const
- std::string **getRawAuthority** () const

  *Returns the raw authority component of this **URI** (p. 2638).*

- std::string **getRawFragment** () const

  *Returns the raw fragment component of this **URI** (p. 2638).*

- std::string **getRawPath** () const

  *Returns the raw path component of this **URI** (p. 2638).*

- std::string **getRawQuery** () const

  *Returns the raw query component of this **URI** (p. 2638).*

- std::string **getRawSchemeSpecificPart** () const

  *Returns the raw scheme-specific part of this **URI** (p. 2638).*

- std::string **getSchemeSpecificPart** () const

  *Returns the decoded scheme-specific part of this **URI** (p. 2638).*

- std::string **getRawUserInfo** () const

  *Returns the raw user-information component of this **URI** (p. 2638).*

- bool **isAbsolute** () const

  *Tells whether or not this **URI** (p. 2638) is absolute.*

- bool **isOpaque** () const

  *Tells whether or not this **URI** (p. 2638) is opaque.*

- **URI normalize** () const

  Normalizes this URI's path.

- **URI parseServerAuthority** () const throw (URISyntaxException)

  Attempts to parse this URI's authority component, if defined, into user-information, host, and port components.

- **URI relativize** (const **URI** &uri) const

  *Relativizes the given **URI** (p. 2638) against this **URI** (p. 2638).*

- **URI resolve** (const std::string &str) const throw (lang::exceptions::IllegalArgumentException)

  *Constructs a new **URI** (p. 2638) by parsing the given string and then resolving it against this **URI** (p. 2638).*

- **URI resolve** (const **URI** &uri) const

  *Resolves the given **URI** (p. 2638) against this **URI** (p. 2638).*

- std::string **toString** () const

  *Returns the content of this **URI** (p. 2638) as a string.*

- **URL toURL** () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)

  *Constructs a **URL** (p. 2677) from this **URI** (p. 2638).*

Static Public Member Functions

- static **URI create** (const std::string uri) throw (lang::exceptions::IllegalArgumentException)

  *Creates a **URI** (p. 2638) by parsing the given string.*

6.571.1 Detailed Description

This class represents an instance of a **URI** (p. 2638) as defined by RFC 2396.

6.571.2 Constructor & Destructor Documentation

6.571.2.1 decaf::net::URI::URI ()

Default Constructor, same as calling a Constructor with all fields empty.

6.571.2.2 decaf::net::URI::URI (const URI & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) as a copy of another **URI** (p. 2638).

Parameters:

uri - uri to copy

6.571.2.3 decaf::net::URI::URI (const std::string & *uri*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) from the given string.

Parameters:

uri - string uri to parse.

6.571.2.4 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *ssp*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) from the given components.

Parameters:

scheme - the uri scheme

ssp - Scheme specific part

fragment - Fragment

6.571.2.5 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *userInfo*, const std::string & *host*, int *port*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) from the given components.

Parameters:

scheme - Scheme name

userInfo - User name and authorization information

host - Host name

port - Port number

path - Path

query - Query

fragment - Fragment

6.571.2.6 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *host*, const std::string & *path*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) from the given components.

Parameters:

scheme - Scheme name

host - Host name

path - Path

fragment - Fragment

6.571.2.7 decaf::net::URI::URI (const std::string & *scheme*, const std::string & *authority*, const std::string & *path*, const std::string & *query*, const std::string & *fragment*) throw (URISyntaxException)

Constructs a **URI** (p. 2638) from the given components.

Parameters:

scheme - Scheme name

authority - Authority

path - Path

query - Query

fragment - Fragment

6.571.2.8 virtual decaf::net::URI::~URI () [inline, virtual]

6.571.3 Member Function Documentation

6.571.3.1 virtual int decaf::net::URI::compareTo (const URI & *value*) const [virtual]

Compares this object with the specified object for order. Returns a negative integer, zero, or a positive integer as this object is less than, equal to, or greater than the specified object.

Parameters:

value - the value to compare to this one.

Returns:

zero if equal minus one if less than and one if greater than.

6.571.3.2 static URI decaf::net::URI::create (const std::string *uri*) throw (lang::exceptions::IllegalArgumentException) [static]

Creates a **URI** (p. 2638) by parsing the given string. This convenience factory method works as if by invoking the URI(string) constructor; any **URISyntaxException** (p. 2665) thrown by the constructor is caught and wrapped in a new IllegalArgumentException object, which is then thrown.

Parameters:

uri - **URI** (p. 2638) string to parse

Exceptions:

IllegalArgumentException

6.571.3.3 virtual bool decaf::net::URI::equals (const URI & *value*) const [virtual]

Returns:

true if this value is considered equal to the passed value.

6.571.3.4 std::string decaf::net::URI::getAuthority () const

Returns:

the decoded authority component of this **URI** (p. 2638).

6.571.3.5 std::string decaf::net::URI::getFragment () const

Returns:

the decoded fragment component of this **URI** (p. 2638).

6.571.3.6 std::string decaf::net::URI::getHost () const

Returns:

the host component of this **URI** (p. 2638).

6.571.3.7 std::string decaf::net::URI::getPath () const

Returns:

the path component of this **URI** (p. 2638).

6.571.3.8 int decaf::net::URI::getPort () const

Returns:

the port component of this **URI** (p. 2638).

6.571.3.9 std::string decaf::net::URI::getQuery () const

Returns:

the query component of this **URI** (p. 2638).

6.571.3.10 std::string decaf::net::URI::getRawAuthority () const

Returns the raw authority component of this **URI** (p. 2638). The authority component of a **URI** (p. 2638), if defined, only contains the commercial-at character ('@') and characters in the unreserved, punct, escaped, and other categories. If the authority is server-based then it is further constrained to have valid user-information, host, and port components.

Returns:

the raw authority component of the **URI** (p. 2638)

6.571.3.11 std::string decaf::net::URI::getRawFragment () const

Returns the raw fragment component of this **URI** (p. 2638). The fragment component of a **URI** (p. 2638), if defined, only contains legal **URI** (p. 2638) characters.

Returns:

the raw fragment component of this **URI** (p. 2638)

6.571.3.12 std::string decaf::net::URI::getRawPath () const

Returns the raw path component of this **URI** (p. 2638). The path component of a **URI** (p. 2638), if defined, only contains the slash character ('/'), the commercial-at character ('@'), and characters in the unreserved, punct, escaped, and other categories.

Returns:

the raw path component of this **URI** (p. 2638)

6.571.3.13 std::string decaf::net::URI::getRawQuery () const

Returns the raw query component of this **URI** (p. 2638). The query component of a **URI** (p. 2638), if defined, only contains legal **URI** (p. 2638) characters.

Returns:

the raw query component of the **URI** (p. 2638).

6.571.3.14 std::string decaf::net::URI::getRawSchemeSpecificPart () const

Returns the raw scheme-specific part of this **URI** (p. 2638). The scheme-specific part is never undefined, though it may be empty. The scheme-specific part of a **URI** (p. 2638) only contains legal **URI** (p. 2638) characters.

Returns:

the raw scheme special part of the uri

6.571.3.15 std::string decaf::net::URI::getRawUserInfo () const

Returns the raw user-information component of this **URI** (p. 2638). The user-information component of a **URI** (p. 2638), if defined, only contains characters in the unreserved, punct, escaped, and other categories.

Returns:

the raw user-information component of the **URI** (p. 2638)

6.571.3.16 std::string decaf::net::URI::getScheme () const

Returns:

the scheme component of this **URI** (p. 2638)

6.571.3.17 std::string decaf::net::URI::getSchemeSpecificPart () const

Returns the decoded scheme-specific part of this **URI** (p. 2638). The string returned by this method is equal to that returned by the getRawSchemeSpecificPart method except that all sequences of escaped octets are decoded.

Returns:

the raw scheme specific part of the uri.

6.571.3.18 std::string decaf::net::URI::getUserInfo () const

Returns:

the user info component of this **URI** (p. 2638)

6.571.3.19 bool decaf::net::URI::isAbsolute () const

Tells whether or not this **URI** (p. 2638) is absolute. A **URI** (p. 2638) is absolute if, and only if, it has a scheme component.

Returns:

true if, and only if, this **URI** (p. 2638) is absolute

6.571.3.20 bool decaf::net::URI::isOpaque () const

Tells whether or not this **URI** (p. 2638) is opaque. A **URI** (p. 2638) is opaque if, and only if, it is absolute and its scheme-specific part does not begin with a slash character ('/'). An opaque **URI** (p. 2638) has a scheme, a scheme-specific part, and possibly a fragment; all other components are undefined.

Returns:

true if, and only if, this **URI** (p. 2638) is opaque

6.571.3.21 URI decaf::net::URI::normalize () const

Normalizes this URI's path. If this **URI** (p. 2638) is opaque, or if its path is already in normal form, then this **URI** (p. 2638) is returned. Otherwise a new **URI** (p. 2638) is constructed that is identical to this **URI** (p. 2638) except that its path is computed by normalizing this URI's path in a manner consistent with RFC 2396, section 5.2, step 6, sub-steps c through f; that is:

1. All "." segments are removed. 2. If a ".." segment is preceded by a non-".." segment then both of these segments are removed. This step is repeated until it is no longer applicable. 3. If the path is relative, and if its first segment contains a colon character (':'), then a "." segment is prepended. This prevents a relative **URI** (p. 2638) with a path such as "a:b/c/d" from later being re-parsed as an opaque **URI** (p. 2638) with a scheme of "a" and a scheme-specific part of "b/c/d". (Deviation from RFC 2396)

A normalized path will begin with one or more ".." segments if there were insufficient non-".." segments preceding them to allow their removal. A normalized path will begin with a "." segment if one was inserted by step 3 above. Otherwise, a normalized path will not contain any "." or ".." segments.

Returns:

A **URI** (p. 2638) equivalent to this **URI** (p. 2638), but whose path is in normal form

6.571.3.22 virtual bool decaf::net::URI::operator< (const URI & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.571.3.23 virtual bool decaf::net::URI::operator== (const URI & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.571.3.24 URI decaf::net::URI::parseServerAuthority () const throw (URISyntaxException)

Attempts to parse this URI's authority component, if defined, into user-information, host, and port components. If this URI's authority component has already been recognized as being server-based then it will already have been parsed into user-information, host, and port components. In this case, or if this **URI** (p. 2638) has no authority component, this method simply returns this **URI** (p. 2638).

Otherwise this method attempts once more to parse the authority component into user-information, host, and port components, and throws an exception describing why the authority component could not be parsed in that way.

Returns:

A **URI** (p. 2638) whose authority field has been parsed as a server-based authority

Exceptions:

URISyntaxException *(p. 2665)* - If the authority component of this **URI** (p. 2638) is defined but cannot be parsed as a server-based authority.

6.571.3.25 URI decaf::net::URI::relativize (const URI & *uri*) const

Relativizes the given **URI** (p. 2638) against this **URI** (p. 2638). The relativization of the given **URI** (p. 2638) against this **URI** (p. 2638) is computed as follows:

1. If either this **URI** (p. 2638) or the given **URI** (p. 2638) are opaque, or if the scheme and authority components of the two URIs are not identical, or if the path of this **URI** (p. 2638) is not a prefix of the path of the given **URI** (p. 2638), then the given **URI** (p. 2638) is returned. 2. Otherwise a new relative hierarchical **URI** (p. 2638) is constructed with query and fragment components taken from the given **URI** (p. 2638) and with a path component computed by removing this URI's path from the beginning of the given URI's path.

Parameters:

uri - The **URI** (p. 2638) to be relativized against this **URI** (p. 2638)

Returns:

The resulting **URI** (p. 2638)

6.571.3.26 URI decaf::net::URI::resolve (const URI & *uri*) const

Resolves the given **URI** (p. 2638) against this **URI** (p. 2638). If the given **URI** (p. 2638) is already absolute, or if this **URI** (p. 2638) is opaque, then a copy of the given **URI** (p. 2638) is returned.

If the given URI's fragment component is defined, its path component is empty, and its scheme, authority, and query components are undefined, then a **URI** (p. 2638) with the given fragment but with all other components equal to those of this **URI** (p. 2638) is returned. This allows a **URI** (p. 2638) representing a standalone fragment reference, such as "#foo", to be usefully resolved against a base **URI** (p. 2638).

Otherwise this method constructs a new hierarchical **URI** (p. 2638) in a manner consistent with RFC 2396, section 5.2; that is:

1. A new **URI** (p. 2638) is constructed with this URI's scheme and the given URI's query and fragment components. 2. If the given **URI** (p. 2638) has an authority component then the new URI's authority and path are taken from the given **URI** (p. 2638). 3. Otherwise the new URI's authority component is copied from this **URI** (p. 2638), and its path is computed as follows:

1. If the given URI's path is absolute then the new URI's path is taken from the given **URI** (p. 2638). 2. Otherwise the given URI's path is relative, and so the new URI's path is computed by resolving the path of the given **URI** (p. 2638) against the path of this **URI** (p. 2638). This is done by concatenating all but the last segment of this URI's path, if any, with the given URI's path and then normalizing the result as if by invoking the normalize method.

The result of this method is absolute if, and only if, either this **URI** (p. 2638) is absolute or the given **URI** (p. 2638) is absolute.

Parameters:

uri - The **URI** (p. 2638) to be resolved against this **URI** (p. 2638)

Returns:

The resulting **URI** (p. 2638)

6.571.3.27 URI decaf::net::URI::resolve (const std::string & *str*) const throw (lang::exceptions::IllegalArgumentException)

Constructs a new **URI** (p. 2638) by parsing the given string and then resolving it against this **URI** (p. 2638). This convenience method works as if invoking it were equivalent to evaluating the expression resolve(URI::create(str)).

Parameters:

str - The string to be parsed into a **URI** (p. 2638)

Returns:

The resulting **URI** (p. 2638)

Exceptions:

IllegalArgumentException - If the given string violates RFC 2396

6.571.3.28 std::string decaf::net::URI::toString () const

Returns the content of this **URI** (p. 2638) as a string. If this **URI** (p. 2638) was created by invoking one of the constructors in this class then a string equivalent to the original input string, or to the string computed from the originally-given components, as appropriate, is returned. Otherwise this **URI** (p. 2638) was created by normalization, resolution, or relativization, and so a string is constructed from this URI's components according to the rules specified in RFC 2396, section 5.2, step 7.

Returns:

the string form of this **URI** (p. 2638)

6.571.3.29 URL decaf::net::URI::toURL () const throw (MalformedURLException, lang::exceptions::IllegalArgumentException)

Constructs a **URL** (p. 2677) from this **URI** (p. 2638). This convenience method works as if invoking it were equivalent to evaluating the expression new **URL** (p. 2677)(this.toString()) after first checking that this **URI** (p. 2638) is absolute.

Returns:

A **URL** (p. 2677) constructed from this **URI** (p. 2638)

Exceptions:

IllegalArgumentException - If this **URL** (p. 2677) is not absolute

MalformedURLException *(p. **1686**)* - If a protocol handler for the **URL** (p. 2677) could not be found, or if some other error occurred while constructing the **URL** (p. 2677)

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URI.h**

6.572 decaf::internal::net::URIEncoderDecoder Class Reference

`#include <src/main/decaf/internal/net/URIEncoderDecoder.h>`

Public Member Functions

- **URIEncoderDecoder** ()
- virtual ~**URIEncoderDecoder** ()

Static Public Member Functions

- static void **validate** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

  Validate a string by checking if it contains any characters other than:.

- static void **validateSimple** (const std::string &s, const std::string &legal) throw (decaf::net::URISyntaxException)

  Validate a string by checking if it contains any characters other than:.

- static std::string **quoteIllegal** (const std::string &s, const std::string &legal)

  All characters except letters ('a'.

- static std::string **encodeOthers** (const std::string &s)

  Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved.

- static std::string **decode** (const std::string &s)

  Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type using the UTF-8 encoding scheme.

6.572.1 Constructor & Destructor Documentation

6.572.1.1 decaf::internal::net::URIEncoderDecoder::URIEncoderDecoder ()

6.572.1.2 virtual decaf::internal::net::URIEncoderDecoder::~URIEncoderDecoder () `[inline, virtual]`

6.572.2 Member Function Documentation

6.572.2.1 static std::string decaf::internal::net::URIEncoderDecoder::decode (const std::string & *s*) `[static]`

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type using the UTF-8 encoding scheme. " and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A%20B%20C %24%25" -> "A B C $%"

Parameters:

s - The encoded string.

Returns:

The decoded version.

6.572.2.2 static std::string decaf::internal::net::URIEncoderDecoder::encodeOthers (const std::string & *s*) [static]

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars are not preserved. They are converted into their hexidecimal value prepended by ".

For example: Euro currency symbol -> "%E2%82%AC".

Parameters:

s - the string to be converted

Returns:

the converted string

6.572.2.3 static std::string decaf::internal::net::URIEncoderDecoder::quoteIllegal (const std::string & *s*, const std::string & *legal*) [static]

All characters except letters ('a'. .'z', 'A'..'Z') and numbers ('0'..'9') and legal characters are converted into their hexidecimal value prepended by ".

For example: '#' -> 23

Other characters, which are chars that are not US-ASCII, and are not ISO Control or are not ISO Space chars, are preserved.

Parameters:

s - the string to be converted

legal - the characters allowed to be preserved in the string s

Returns:

converted string

6.572.2.4 static void decaf::internal::net::URIEncoderDecoder::validate (const std::string & *s*, const std::string & *legal*) throw (decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than:. 1. letters ('a'..'z', 'A'..'Z') 2. numbers ('0'..'9') 3. characters in the legalset parameter 4. characters that are not ISO Control or are not ISO Space characters)

Parameters:

s - the string to be validated

legal - the characters allowed in the string s

6.572.2.5 static void decaf::internal::net::URIEncoderDecoder::validateSimple (const std::string & *s*, const std::string & *legal*) throw (decaf::net::URISyntaxException) [static]

Validate a string by checking if it contains any characters other than:. 1. letters ('a'..'z', 'A'..'Z') 2. numbers ('0'..'9') 3. characters in the legalset parameter

Parameters:

s - the string to be validated

legal - the characters allowed in the string s

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIEncoderDecoder.h**

6.573 decaf::internal::net::URIHelper Class Reference

Helper class used by the URI classes in encoding and decoding of URI's.

`#include <src/main/decaf/internal/net/URIHelper.h>`

Public Member Functions

- **URIHelper** (const std::string &unreserved, const std::string &punct, const std::string &reserved, const std::string &someLegal, const std::string &allLegal)

  *Setup the **URIHelper** (p. 2652) with values assigned to the various fields that are used in the validation process.*

- **URIHelper** ()

  Sets up the filter strings with sane defaults.

- virtual ~**URIHelper** ()
- **URIType parseURI** (const std::string &uri, bool forceServer) throw (decaf::net::URISyntaxException)

  Parse the passed in URI.

- void **validateScheme** (const std::string &uri, const std::string &scheme, int index) throw (decaf::net::URISyntaxException)

  Validate the schema portin of the URI.

- void **validateSsp** (const std::string &uri, const std::string &ssp, std::size_t index) throw (decaf::net::URISyntaxException)

  Validate that the URI Ssp Segment contains no invalid encodings.

- void **validateAuthority** (const std::string &uri, const std::string &authority, std::size_t index) throw (decaf::net::URISyntaxException)

  Validate that the URI Authority Segment contains no invalid encodings.

- void **validatePath** (const std::string &uri, const std::string &path, std::size_t index) throw (decaf::net::URISyntaxException)

  Validate that the URI Path Segment contains no invalid encodings.

- void **validateQuery** (const std::string &uri, const std::string &query, std::size_t index) throw (decaf::net::URISyntaxException)

  Validate that the URI Query Segment contains no invalid encodings.

- void **validateFragment** (const std::string &uri, const std::string &fragment, std::size_t index) throw (decaf::net::URISyntaxException)

  Validate that the URI fragment contains no invalid encodings.

- **URIType parseAuthority** (bool forceServer, const std::string &authority) throw (decaf::net::URISyntaxException)

  determine the host, port and user-info if the authority parses successfully to a server based authority

- void **validateUserinfo** (const std::string &uri, const std::string &userinfo, std::size_t index) throw (decaf::net::URISyntaxException)

  Check the supplied user info for validity.

- bool **isValidHost** (bool forceServer, const std::string &host) throw (decaf::net::URISyntaxException)

  distinguish between IPv4, IPv6, domain name and validate it based on its type

- bool **isValidDomainName** (const std::string &host)

  Validates the string past to determine if it is a well formed domain name.

- bool **isValidIPv4Address** (const std::string &host)

  Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9.

- bool **isValidIP6Address** (const std::string &ipAddress)

  Determines if the given address is valid according to the IPv6 spec.

- bool **isValidIP4Word** (const std::string &word)

  Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

- bool **isValidHexChar** (char c)

  Determines if the given char is a valid Hex char.

6.573.1 Detailed Description

Helper class used by the URI classes in encoding and decoding of URI's.

6.573.2 Constructor & Destructor Documentation

6.573.2.1 decaf::internal::net::URIHelper::URIHelper (const std::string & *unreserved*, const std::string & *punct*, const std::string & *reserved*, const std::string & *someLegal*, const std::string & *allLegal*)

Setup the **URIHelper** (p. 2652) with values assigned to the various fields that are used in the validation process. The defaults are overridden by these values.

Parameters:

unreserved - characters not reserved for use.

punct - allowable punctuation symbols.

reserved - characters not allowed for general use in the URI.

someLegal - characters that are legal in certain cases.

allLegal - characters that are always legal.

6.573.2.2 decaf::internal::net::URIHelper::URIHelper ()

Sets up the filter strings with sane defaults.

6.573.2.3 virtual decaf::internal::net::URIHelper::~URIHelper () [inline, virtual]

6.573.3 Member Function Documentation

6.573.3.1 bool decaf::internal::net::URIHelper::isValidDomainName (const std::string & *host*)

Validates the string past to determine if it is a well formed domain name.

Parameters:

host - domain name to validate.

Returns:

true if host is well formed.

6.573.3.2 bool decaf::internal::net::URIHelper::isValidHexChar (char *c*)

Determines if the given char is a valid Hex char. Valid chars are A-F (upper or lower case) and 0-9.

Parameters:

c - char to inspect

Returns:

true if c is a valid hex char.

6.573.3.3 bool decaf::internal::net::URIHelper::isValidHost (bool *forceServer*, const std::string & *host*) throw (decaf::net::URISyntaxException)

distinguish between IPv4, IPv6, domain name and validate it based on its type

Parameters:

forceServer - true if the forceServer mode should be active.

host - Host string to validate.

Returns:

true if the host value if a valid domain name.

Exceptions:

URISyntaxException if the host is invalid and forceServer is true.

6.573.3.4 bool decaf::internal::net::URIHelper::isValidIP4Word (const std::string & *word*)

Check is the string passed contains a Valid IPv4 word, which is an integer in the range of 0 to 255.

Parameters:

word - string value to check.

Returns:

true if the word is a valid IPv4 word.

6.573.3.5 bool decaf::internal::net::URIHelper::isValidIP6Address (const std::string & *ipAddress*)

Determines if the given address is valid according to the IPv6 spec.

Parameters:

ipAddress - string ip address value to validate.

Returns:

true if the address string is valid.

6.573.3.6 bool decaf::internal::net::URIHelper::isValidIPv4Address (const std::string & *host*)

Validate if the host value is a well formed IPv4 address, this is the form XXX.XXX.XXX.XXX were X is any number 0-9. and XXX is not greater than 255.

Parameters:

host - IPv4 address string to parse.

Returns:

true if host is a well formed IPv4 address.

6.573.3.7 URIType decaf::internal::net::URIHelper::parseAuthority (bool *forceServer*, const std::string & *authority*) throw (decaf::net::URISyntaxException)

determine the host, port and user-info if the authority parses successfully to a server based authority behavior in error cases: if forceServer is true, throw URISyntaxException with the proper diagnostic messages. if forceServer is false assume this is a registry based uri, and just return leaving the host, port and user-info fields undefined.

and there are some error cases where URISyntaxException is thrown regardless of the forceServer parameter e.g. mal-formed ipv6 address

Parameters:

forceServer

authority

Returns:

a **URIType** (p. 2669) instance containing the parsed data.

Exceptions:

URISyntaxException

6.573.3.8 URIType decaf::internal::net::URIHelper::parseURI (const std::string & *uri*, bool *forceServer*) throw (decaf::net::URISyntaxException)

Parse the passed in URI.

Parameters:

uri - the URI to Parse

forceServer - if true invalid URI data throws an Exception

Returns:

a **URIType** (p. 2669) instance containing the parsed data.

Exceptions:

URISyntaxException if forceServer is true and the URI is invalid.

6.573.3.9 void decaf::internal::net::URIHelper::validateAuthority (const std::string & *uri*, const std::string & *authority*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Authority Segment contains no invalid encodings.

Parameters:

uri - the full uri.

authority - the Authority to check.

index - position in the uri where Authority starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.10 void decaf::internal::net::URIHelper::validateFragment (const std::string & *uri*, const std::string & *fragment*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI fragment contains no invalid encodings.

Parameters:

uri - the full uri.

fragment - the fragment to check.

index - position in the uri where fragment starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.11 void decaf::internal::net::URIHelper::validatePath (const std::string & *uri*, const std::string & *path*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Path Segment contains no invalid encodings.

Parameters:

uri - the full uri.

path - the path to check.

index - position in the uri where path starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.12 void decaf::internal::net::URIHelper::validateQuery (const std::string & *uri*, const std::string & *query*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Query Segment contains no invalid encodings.

Parameters:

uri - the full uri.

query - the query to check.

index - position in the uri where fragment starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.13 void decaf::internal::net::URIHelper::validateScheme (const std::string & *uri*, const std::string & *scheme*, int *index*) throw (decaf::net::URISyntaxException)

Validate the schema portin of the URI.

Parameters:

uri - the URI to check.

scheme - the schema section of the URI.

index - index in uri where schema starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.14 void decaf::internal::net::URIHelper::validateSsp (const std::string & *uri*, const std::string & *ssp*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Validate that the URI Ssp Segment contains no invalid encodings.

Parameters:

uri - the full uri.

ssp - the SSP to check.

index - position in the uri where Ssp starts.

Exceptions:

URISyntaxException if the fragment has errors.

6.573.3.15 void decaf::internal::net::URIHelper::validateUserinfo (const std::string & *uri*, const std::string & *userinfo*, std::size_t *index*) throw (decaf::net::URISyntaxException)

Check the supplied user info for validity.

Parameters:

uri - the uri to parse.

userinfo - supplied user info

index - index into the URI string where the data is located.

Returns:

true if valid

Exceptions:

URISyntaxException if an error occurs

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIHelper.h**

6.574 activemq::transport::failover::URIPool Class Reference

`#include <src/main/activemq/transport/failover/URIPool.h>`

Public Member Functions

- **URIPool** ()

  Create an Empty URI Pool.

- **URIPool** (const **decaf::util::List**< **URI** > &uris)

  Creates a new URI Pool using the given list as the initial Free List.

- virtual ~**URIPool** ()
- **URI getURI** () throw (decaf::lang::exceptions::NoSuchElementException)

  Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException.

- void **addURI** (const **URI** &uri)

  Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

- void **addURIs** (const **StlList**< **URI** > &uris)

  Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

- void **removeURI** (const **URI** &uri)

  Remove a given URI from the Free List.

- bool **isRandomize** () const

  Is the URI that is given randomly picked from the pool or is each one taken in sequence.

- void **setRandomize** (bool value)

  Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

6.574.1 Constructor & Destructor Documentation

6.574.1.1 activemq::transport::failover::URIPool::URIPool ()

Create an Empty URI Pool.

6.574.1.2 activemq::transport::failover::URIPool::URIPool (const decaf::util::List< URI > & *uris*)

Creates a new URI Pool using the given list as the initial Free List.

Parameters:

uris - List of URI to place in the Pool.

6.574.1.3 virtual activemq::transport::failover::URIPool::~URIPool () [virtual]

6.574.2 Member Function Documentation

6.574.2.1 void activemq::transport::failover::URIPool::addURI (const URI & *uri*)

Adds a URI to the free list, callers that have previously taken one using the `getURI` method should always return the URI when they close the resource that was connected to that URI.

Parameters:

uri - a URI previously taken from the pool.

6.574.2.2 void activemq::transport::failover::URIPool::addURIs (const StlList< URI > & *uris*)

Adds a List of URIs to this Pool, the method checks for duplicates already in the pool and does not add those.

Parameters:

uris - List of URIs to add into the Pool.

6.574.2.3 URI activemq::transport::failover::URIPool::getURI () throw (decaf::lang::exceptions::NoSuchElementException)

Fetches the next available URI from the pool, if there are no more URIs free when this method is called it throws a NoSuchElementException. Receiving the exception is not an indication that a URI won't be available in the future, the caller should react accordingly.

Returns:

the next free URI in the Pool.

Exceptions:

NoSuchElementException if there are none free currently.

6.574.2.4 bool activemq::transport::failover::URIPool::isRandomize () const [inline]

Is the URI that is given randomly picked from the pool or is each one taken in sequence.

Returns:

true if URI gets are random.

6.574.2.5 void activemq::transport::failover::URIPool::removeURI (const URI & *uri*)

Remove a given URI from the Free List.

Parameters:

uri - the URI to find and remove from the free list

6.574.2.6 void activemq::transport::failover::URIPool::setRandomize (bool *value*) [inline]

Sets if the URI's that are taken from the pool are chosen Randomly or are taken in the order they are in the list.

Parameters:

value - true indicates URI gets are random.

The documentation for this class was generated from the following file:

- src/main/activemq/transport/failover/**URIPool.h**

6.575 activemq::util::URISupport Class Reference

`#include <src/main/activemq/util/URISupport.h>`

Static Public Member Functions

- static void **parseURL** (const std::string &**URI**, **decaf::util::Properties** &properties) throw (decaf::lang::exceptions::IllegalArgumentException)

  Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

- static **CompositeData parseComposite** (const **URI** &uri) throw (decaf::net::URISyntaxException)

  Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

- static **decaf::util::Properties parseQuery** (std::string query) throw (decaf::lang::exceptions::IllegalArgumentException)

  Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static void **parseQuery** (std::string query, **decaf::util::Properties** *properties) throw (decaf::lang::exceptions::IllegalArgumentException)

  Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

- static std::string **createQueryString** (const **Properties** &options) throw (decaf::net::URISyntaxException)

  Given a properties object create a string that can be appended to a URI as a valid Query string.

6.575.1 Member Function Documentation

6.575.1.1 static std::string activemq::util::URISupport::createQueryString (const Properties & *options*) throw (decaf::net::URISyntaxException) [static]

Given a properties object create a string that can be appended to a URI as a valid Query string.

Parameters:

options Properties object containing key / value query values.

Returns:

a valid URI query string.

Exceptions:

URISyntaxException if the string in the Properties object can't be encoded into a valid URI Query string.

6.575.1.2 static CompositeData activemq::util::URISupport::parseComposite (const URI & *uri*) throw (decaf::net::URISyntaxException) [static]

Parses a Composite URI into a Composite Data instance, the Composite URI takes the for scheme://(uri1,uri2,...uriN)?param1=value1, each of the composite URIs is stored in the CompositeData's internal list.

Parameters:

uri - The Composite URI to parse.

Returns:

a new **CompositeData** (p. 904) object with the parsed data

Exceptions:

URISyntaxException if the URI is not well formed.

6.575.1.3 static void activemq::util::URISupport::parseQuery (std::string *query*, decaf::util::Properties * *properties*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters:

query - the query string to parse.

properties - object pointer to get the parsed output.

Exceptions:

IllegalArgumentException if the Query string is not well formed.

6.575.1.4 static decaf::util::Properties activemq::util::URISupport::parseQuery (std::string *query*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parse the Query portion of a URI String and return a Simple Properties object containing the parameter names as keys, and the parameter values and values of the Properties.

Parameters:

query The query string to parse and extract the encoded properties.

Returns:

Properties object with the parsed output.

Exceptions:

IllegalArgumentException if the Query string is not well formed.

6.575.1.5 static void activemq::util::URISupport::parseURL (const std::string & *URI*, decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalArgumentException) [static]

Parses the properties out of the provided Broker URI and sets them in the passed Properties Object.

Parameters:

URI a Broker URI to parse

properties a Properties object to set the parsed values in

Exceptions:

IllegalArgumentException if the passed URI is invalid

The documentation for this class was generated from the following file:

- src/main/activemq/util/**URISupport.h**

6.576 decaf::net::URISyntaxException Class Reference

`#include <src/main/decaf/net/URISyntaxException.h>`Inheritance diagram for decaf::net::URISyntaxException:

Public Member Functions

- **URISyntaxException** () throw ()
  Default Constructor.
- **URISyntaxException** (const Exception &ex) throw ()
  Conversion Constructor from some other Exception.
- **URISyntaxException** (const **URISyntaxException** &ex) throw ()
  Copy Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const std::exception *__cause__, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const std::exception *__cause__) throw ()
  Constructor.
- **URISyntaxException** (const char *file, const int lineNumber, const char *msg DECAF_UNUSED) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **URISyntaxException** (const char *file, const int lineNumber, const std::string &input, const std::string &reason, std::size_t index) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- virtual **URISyntaxException** * **clone** () const
  Clones this exception.
- virtual ~**URISyntaxException** () throw ()
- std::string **getInput** () const
- std::string **getReason** () const
- std::size_t **getIndex** () const

6.576.1 Constructor & Destructor Documentation

6.576.1.1 decaf::net::URISyntaxException::URISyntaxException () throw () [inline]

Default Constructor.

6.576.1.2 decaf::net::URISyntaxException::URISyntaxException (const Exception & *ex*) throw () [inline]

Conversion Constructor from some other Exception.

Parameters:

ex An exception that should become this type of Exception

6.576.1.3 decaf::net::URISyntaxException::URISyntaxException (const URISyntaxException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex An exception that should become this type of Exception

6.576.1.4 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.576.1.5 decaf::net::URISyntaxException::URISyntaxException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.576.1.6 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const char *msg *DECAF_UNUSED*) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.576.1.7 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::string & *input*, const std::string & *reason*) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

Parameters:

file The file name where exception occurs.

lineNumber The line number where the exception occurred.

input The **URL** (p. 2677) that caused the exception.

reason The reason for the failure.

6.576.1.8 decaf::net::URISyntaxException::URISyntaxException (const char * *file*, const int *lineNumber*, const std::string & *input*, const std::string & *reason*, std::size_t *index*) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the input string that caused the error and the reason for the error.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

input The input **URI** (p. 2638) that caused the exception

reason The reason for the failure.

index The index in the **URI** (p. 2638) string where the error occurred.

6.576.1.9 virtual decaf::net::URISyntaxException::~URISyntaxException () throw () [inline, virtual]

6.576.2 Member Function Documentation

6.576.2.1 virtual URISyntaxException* decaf::net::URISyntaxException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

a new Exception instance that is a copy of this Exception object.

Reimplemented from **decaf::lang::Exception** (p. 1271).

6.576.2.2 std::size_t decaf::net::URISyntaxException::getIndex () const [inline]

Returns:

the index in the input string where the error occured or -1

6.576.2.3 std::string decaf::net::URISyntaxException::getInput () const [inline]

Returns:

the Input string that cause this exception or ""

6.576.2.4 std::string decaf::net::URISyntaxException::getReason () const [inline]

Returns:

the Reason given for this failure, or ""

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URISyntaxException.h**

6.577 decaf::internal::net::URIType Class Reference

Basic type object that holds data that composes a given URI.

`#include <src/main/decaf/internal/net/URIType.h>`

Public Member Functions

- **URIType** (const std::string &source)
- **URIType** ()
- virtual ~**URIType** ()
- std::string **getSource** () const

  *Gets the source URI string that was parsed to obtain this **URIType** (p. 2669) instance and the resulting data,.*

- void **setSource** (const std::string &source)

  *Sets the source URI string that was parsed to obtain this **URIType** (p. 2669) instance and the resulting data,.*

- std::string **getScheme** () const

  Gets the Scheme of the URI, e.g.

- void **setScheme** (const std::string &scheme)

  Sets the Scheme of the URI, e.g.

- std::string **getSchemeSpecificPart** () const

  Gets the Scheme Specific Part of the URI.

- void **setSchemeSpecificPart** (const std::string &schemeSpecificPart)

  Sets the Scheme Specific Part of the URI.

- std::string **getAuthority** () const

  Gets the Authority of the URI.

- void **setAuthority** (const std::string &authority)

  Sets the Authority of the URI.

- std::string **getUserInfo** () const

  Gets the user info part of the URI, e.g.

- void **setUserInfo** (const std::string &userinfo)

  Sets the user info part of the URI, e.g.

- std::string **getHost** () const

  Gets the Host name part of the URI.

- void **setHost** (const std::string &host)

  Sets the Host name part of the URI.

- int **getPort** () const

Gets the port part of the URI.

- void **setPort** (int port)

  Sets the port part of the URI.

- std::string **getPath** () const

  Gets the Path part of the URI.

- void **setPath** (const std::string &path)

  Sets the Path part of the URI.

- std::string **getQuery** () const

  Gets the Query part of the URI.

- void **setQuery** (const std::string &query)

  Sets the Query part of the URI.

- std::string **getFragment** () const

  Gets the Fragment part of the URI.

- void **setFragment** (const std::string &fragment)

  Sets the Fragment part of the URI.

- bool **isOpaque** () const

  Gets if the URI is Opaque.

- void **setOpaque** (bool opaque)

  Sets if the URI is Opaque.

- bool **isAbsolute** () const

  Gets if the URI is Absolute.

- void **setAbsolute** (bool absolute)

  Sets if the URI is Absolute.

- bool **isServerAuthority** () const

  Gets if the URI is a Server Authority.

- void **setServerAuthority** (bool serverAuthority)

  Sets if the URI is a Server Authority.

- bool **isValid** () const

  Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

- void **setValid** (bool valid)

  Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

6.577.1 Detailed Description

Basic type object that holds data that composes a given URI.

6.577.2 Constructor & Destructor Documentation

6.577.2.1 decaf::internal::net::URIType::URIType (const std::string & *source*) [inline]

6.577.2.2 decaf::internal::net::URIType::URIType () [inline]

6.577.2.3 virtual decaf::internal::net::URIType::~URIType () [inline, virtual]

6.577.3 Member Function Documentation

6.577.3.1 std::string decaf::internal::net::URIType::getAuthority () const [inline]

Gets the Authority of the URI.

Returns:

Authority part string.

6.577.3.2 std::string decaf::internal::net::URIType::getFragment () const [inline]

Gets the Fragment part of the URI.

Returns:

Fragment part string.

6.577.3.3 std::string decaf::internal::net::URIType::getHost () const [inline]

Gets the Host name part of the URI.

Returns:

Host name part string.

6.577.3.4 std::string decaf::internal::net::URIType::getPath () const [inline]

Gets the Path part of the URI.

Returns:

Path part string.

6.577.3.5 int decaf::internal::net::URIType::getPort () const [inline]

Gets the port part of the URI.

Returns:

port part string, -1 if not set.

6.577.3.6 std::string decaf::internal::net::URIType::getQuery () const [inline]

Gets the Query part of the URI.

Returns:

Query part string.

6.577.3.7 std::string decaf::internal::net::URIType::getScheme () const [inline]

Gets the Scheme of the URI, e.g. scheme ("http"/"ftp"/...).

Returns:

scheme part string.

6.577.3.8 std::string decaf::internal::net::URIType::getSchemeSpecificPart () const [inline]

Gets the Scheme Specific Part of the URI.

Returns:

scheme specific part string.

6.577.3.9 std::string decaf::internal::net::URIType::getSource () const [inline]

Gets the source URI string that was parsed to obtain this **URIType** (p. 2669) instance and the resulting data,.

Returns:

the source URI string

6.577.3.10 std::string decaf::internal::net::URIType::getUserInfo () const [inline]

Gets the user info part of the URI, e.g. user name, as in `http://user:passwd@host:port/`

Returns:

user info part string.

6.577.3.11 bool decaf::internal::net::URIType::isAbsolute () const [inline]

Gets if the URI is Absolute.

Returns:

true if Absolute.

6.577.3.12 bool decaf::internal::net::URIType::isOpaque () const [inline]

Gets if the URI is Opaque.

Returns:

true if opaque.

6.577.3.13 bool decaf::internal::net::URIType::isServerAuthority () const [inline]

Gets if the URI is a Server Authority.

Returns:

true if Server Authority.

6.577.3.14 bool decaf::internal::net::URIType::isValid () const [inline]

Gets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Returns:

true if the **URIType** (p. 2669) contains valid data.

6.577.3.15 void decaf::internal::net::URIType::setAbsolute (bool *absolute*) [inline]

Sets if the URI is Absolute.

Parameters:

absolute - true if Absolute.

6.577.3.16 void decaf::internal::net::URIType::setAuthority (const std::string & *authority*) [inline]

Sets the Authority of the URI.

Parameters:

authority Authority part string.

6.577.3.17 void decaf::internal::net::URIType::setFragment (const std::string & *fragment*) [inline]

Sets the Fragment part of the URI.

Parameters:

fragment - Fragment part string.

6.577.3.18 void decaf::internal::net::URIType::setHost (const std::string & *host*) [inline]

Sets the Host name part of the URI.

Parameters:

host - Host name part string.

6.577.3.19 void decaf::internal::net::URIType::setOpaque (bool *opaque*) [inline]

Sets if the URI is Opaque.

Parameters:

opaque true if opaque.

6.577.3.20 void decaf::internal::net::URIType::setPath (const std::string & *path*) [inline]

Sets the Path part of the URI.

Parameters:

path - Path part string.

6.577.3.21 void decaf::internal::net::URIType::setPort (int *port*) [inline]

Sets the port part of the URI.

Parameters:

port - port part string, -1 if not set.

6.577.3.22 void decaf::internal::net::URIType::setQuery (const std::string & *query*) [inline]

Sets the Query part of the URI.

Parameters:

query - Query part string.

6.577.3.23 void decaf::internal::net::URIType::setScheme (const std::string & *scheme*) [inline]

Sets the Scheme of the URI, e.g. scheme ("http"/"ftp"/...).

Parameters:

scheme - scheme part string.

6.577.3.24 void decaf::internal::net::URIType::setSchemeSpecificPart (const std::string & *schemeSpecificPart*) [inline]

Sets the Scheme Specific Part of the URI.

Parameters:

schemeSpecificPart - scheme specific part string.

6.577.3.25 void decaf::internal::net::URIType::setServerAuthority (bool *serverAuthority*) [inline]

Sets if the URI is a Server Authority.

Parameters:

serverAuthority - true if Server Authority.

6.577.3.26 void decaf::internal::net::URIType::setSource (const std::string & *source*) [inline]

Sets the source URI string that was parsed to obtain this **URIType** (p. 2669) instance and the resulting data,.

Parameters:

source - the source URI string

6.577.3.27 void decaf::internal::net::URIType::setUserInfo (const std::string & *userinfo*) [inline]

Sets the user info part of the URI, e.g. user name, as in `http://user:passwd@host:port/`

Parameters:

userinfo - user info part string.

6.577.3.28 void decaf::internal::net::URIType::setValid (bool *valid*) [inline]

Sets if the URI is valid, meaning that the source has been set and parsed and all relevant data fields have been set.

Parameters:

valid - true if the **URIType** (p. 2669) contains valid data.

The documentation for this class was generated from the following file:

- src/main/decaf/internal/net/**URIType.h**

6.578 decaf::net::URL Class Reference

Class **URL** (p. 2677) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.

`#include <src/main/decaf/net/URL.h>`

Public Member Functions

- **URL** ()
- **URL** (const std::string &url)
- virtual ~**URL** ()

6.578.1 Detailed Description

Class **URL** (p. 2677) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web. A resource can be something as simple as a file or a directory, or it can be a reference to a more complicated object, such as a query to a database or to a search engine. More information on the types of URLs and their formats can be found at:

`http://www.ksc.nasa.gov/facts/internet/url-primer.html`

In general, a **URL** (p. 2677) can be broken into several parts. The previous example of a **URL** (p. 2677) indicates that the protocol to use is http (HyperText Transfer Protocol) and that the information resides on a host machine named www.ksc.nasa.gov. The information on that host machine is named /facts/internet/url-primer.html. The exact meaning of this name on the host machine is both protocol dependent and host dependent. The information normally resides in a file, but it could be generated on the fly. This component of the **URL** (p. 2677) is called the path component.

A **URL** (p. 2677) can optionally specify a "port", which is the port number to which the TCP connection is made on the remote host machine. If the port is not specified, the default port for the protocol is used instead. For example, the default port for http is 80. An alternative port could be specified as:

`http://www.ksc.nasa.gov:80/facts/internet/url-primer.html`

The syntax of **URL** (p. 2677) is defined by RFC 2396: Uniform Resource Identifiers (**URI** (p. 2638)): Generic Syntax, amended by RFC 2732: Format for Literal IPv6 Addresses in URLs. The Literal IPv6 address format also supports scope_ids. The syntax and usage of scope_ids is described here.

A **URL** (p. 2677) may have appended to it a "fragment", also known as a "ref" or a "reference". The fragment is indicated by the sharp sign character "#" followed by more characters. For example,

`http://www.apache.org/cms/index.html#chapter1`

This fragment is not technically part of the **URL** (p. 2677). Rather, it indicates that after the specified resource is retrieved, the application is specifically interested in that part of the document that has the tag chapter1 attached to it. The meaning of a tag is resource specific.

An application can also specify a "relative URL", which contains only enough information to reach the resource relative to another **URL** (p. 2677). Relative URLs are frequently used within HTML pages. For example, if the contents of the **URL** (p. 2677):

`http://www.apache.org/cms/index.html`

contained within it the relative **URL** (p. 2677):

FAQ.html

it would be a shorthand for:

`http://www.apache.org/cms/FAQ.html`

The relative **URL** (p. 2677) need not specify all the components of a **URL** (p. 2677). If the protocol, host name, or port number is missing, the value is inherited from the fully specified **URL** (p. 2677). The file component must be specified. The optional fragment is not inherited.

The **URL** (p. 2677) class does not itself encode or decode any **URL** (p. 2677) components according to the escaping mechanism defined in RFC2396. It is the responsibility of the caller to encode any fields, which need to be escaped prior to calling **URL** (p. 2677), and also to decode any escaped fields, that are returned from **URL** (p. 2677). Furthermore, because **URL** (p. 2677) has no knowledge of **URL** (p. 2677) escaping, it does not recognise equivalence between the encoded or decoded form of the same **URL** (p. 2677). For example, the two URLs:

`http://foo.com/hello` world/ and `http://foo.com/hello%20world`

would be considered not equal to each other.

Note, the **URI** (p. 2638) class does perform escaping of its component fields in certain circumstances. The recommended way to manage the encoding and decoding of URLs is to use **URI** (p. 2638), and to convert between these two classes using toURI() and **URI.toURL()** (p. 2648).

The **URLEncoder** (p. 2680) and **URLDecoder** (p. 2679) classes can also be used, but only for HTML form encoding, which is not the same as the encoding scheme defined in RFC2396.

6.578.2 Constructor & Destructor Documentation

6.578.2.1 decaf::net::URL::URL ()

6.578.2.2 decaf::net::URL::URL (const std::string & *url*)

6.578.2.3 virtual decaf::net::URL::~URL () `[inline, virtual]`

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URL.h**

6.579 decaf::net::URLDecoder Class Reference

`#include <src/main/decaf/net/URLDecoder.h>`

Public Member Functions

- virtual ~**URLDecoder** ()

Static Public Member Functions

- static std::string **decode** (const std::string &value)

  Decodes the string argument which is assumed to be encoded in the ***x-www-form-urlencoded*** *MIME content type.*

6.579.1 Constructor & Destructor Documentation

6.579.1.1 virtual decaf::net::URLDecoder::~URLDecoder () [inline, virtual]

6.579.2 Member Function Documentation

6.579.2.1 static std::string decaf::net::URLDecoder::decode (const std::string & *value*) [static]

Decodes the string argument which is assumed to be encoded in the `x-www-form-urlencoded` MIME content type. '+' will be converted to space, '' and two following hex digit characters are converted to the equivalent byte value. All other characters are passed through unmodified.

e.g. "A+B+C %24%25" -> "A B C $%"

Parameters:

value - string The encoded string.

Returns:

The decoded version as a string.

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URLDecoder.h**

6.580 decaf::net::URLEncoder Class Reference

`#include <src/main/decaf/net/URLEncoder.h>`

Public Member Functions

- virtual ~**URLEncoder** ()

Static Public Member Functions

- static std::string **encode** (const std::string &value)

  This class contains a utility method for converting a string to the format required by the ***application/x-www-form-urlencoded*** *MIME content type.*

6.580.1 Constructor & Destructor Documentation

6.580.1.1 virtual decaf::net::URLEncoder::~URLEncoder () [inline, virtual]

6.580.2 Member Function Documentation

6.580.2.1 static std::string decaf::net::URLEncoder::encode (const std::string & *value*) [static]

This class contains a utility method for converting a string to the format required by the `application/x-www-form-urlencoded` MIME content type. All characters except letters ('a'..'z', 'A'..'Z') and numbers ('0'..'9') and characters '.', '-', '*', '_' are converted into their hexidecimal value prepended by ''.

For example: '#' -> 23

In addition, spaces are substituted by '+'

Parameters:

value - the string to be converted

Returns:

the converted string

The documentation for this class was generated from the following file:

- src/main/decaf/net/**URLEncoder.h**

6.581 activemq::util::Usage Class Reference

`#include <src/main/activemq/util/Usage.h>`Inheritance diagram for activemq::util::Usage:

Public Member Functions

- virtual ~**Usage** ()
- virtual void **waitForSpace** ()=0

  *Waits forever for more space to be returned to this **Usage** (p. 2681) Manager.*

- virtual void **waitForSpace** (unsigned int timeout)=0

  *Waits for more space to be returned to this **Usage** (p. 2681) Manager, times out when the given time span in milliseconds elapses.*

- virtual void **enqueueUsage** (unsigned long long value)=0

  Tries to increase the usage by value amount but blocks if this object is currently full.

- virtual void **increaseUsage** (unsigned long long value)=0

  Increases the usage by the value amount.

- virtual void **decreaseUsage** (unsigned long long value)=0

  Decreases the usage by the value amount.

- virtual bool **isFull** () const =0

  *Returns true if this **Usage** (p. 2681) instance is full, i.e.*

6.581.1 Constructor & Destructor Documentation

6.581.1.1 virtual activemq::util::Usage::~Usage () `[inline, virtual]`

6.581.2 Member Function Documentation

6.581.2.1 virtual void activemq::util::Usage::decreaseUsage (unsigned long long *value*) `[pure virtual]`

Decreases the usage by the value amount.

Parameters:

value Amount of space to return to the pool

Implemented in **activemq::util::MemoryUsage** (p. 1734).

6.581.2.2 virtual void activemq::util::Usage::enqueueUsage (unsigned long long *value*) `[pure virtual]`

Tries to increase the usage by value amount but blocks if this object is currently full.

Parameters:

value Amount of usage in bytes to add.

Implemented in **activemq::util::MemoryUsage** (p. 1734).

6.581.2.3 virtual void activemq::util::Usage::increaseUsage (unsigned long long *value*) [pure virtual]

Increases the usage by the value amount.

Parameters:

value Amount of usage to add.

Implemented in **activemq::util::MemoryUsage** (p. 1735).

6.581.2.4 virtual bool activemq::util::Usage::isFull () const [pure virtual]

Returns true if this **Usage** (p. 2681) instance is full, i.e. **Usage** (p. 2681) >= 100%

Returns:

true if **Usage** (p. 2681) is at the Full point.

Implemented in **activemq::util::MemoryUsage** (p. 1735).

6.581.2.5 virtual void activemq::util::Usage::waitForSpace (unsigned int *timeout*) [pure virtual]

Waits for more space to be returned to this **Usage** (p. 2681) Manager, times out when the given time span in milliseconds elapses.

Parameters:

timeout The time to wait for more space.

Implemented in **activemq::util::MemoryUsage** (p. 1735).

6.581.2.6 virtual void activemq::util::Usage::waitForSpace () [pure virtual]

Waits forever for more space to be returned to this **Usage** (p. 2681) Manager.

Implemented in **activemq::util::MemoryUsage** (p. 1736).

The documentation for this class was generated from the following file:

- src/main/activemq/util/**Usage.h**

6.582 decaf::io::UTFDataFormatException Class Reference

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

`#include <src/main/decaf/io/UTFDataFormatException.h>`Inheritance diagram for decaf::io::UTFDataFormatException:

Public Member Functions

- **UTFDataFormatException** () throw ()
  Default Constructor.
- **UTFDataFormatException** (const **lang::Exception** &ex) throw ()
  Copy Constructor.
- **UTFDataFormatException** (const **UTFDataFormatException** &ex) throw ()
  Copy Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const std::exception ***cause**, const char *msg,...) throw ()
  Constructor - Initializes the file name and line number where this message occurred.
- **UTFDataFormatException** (const std::exception ***cause**) throw ()
  Constructor.
- **UTFDataFormatException** (const char *file, const int lineNumber, const char *msg,...) throw ()
  Constructor.
- virtual **UTFDataFormatException** * **clone** () const
  Clones this exception.
- virtual ~**UTFDataFormatException** () throw ()

6.582.1 Detailed Description

Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Since:

1.0

6.582.2 Constructor & Destructor Documentation

6.582.2.1 decaf::io::UTFDataFormatException::UTFDataFormatException () throw () [inline]

Default Constructor.

6.582.2.2 decaf::io::UTFDataFormatException::UTFDataFormatException (const lang::Exception & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy

6.582.2.3 decaf::io::UTFDataFormatException::UTFDataFormatException (const UTFDataFormatException & *ex*) throw () [inline]

Copy Constructor.

Parameters:

ex the exception to copy, which is an instance of this type

6.582.2.4 decaf::io::UTFDataFormatException::UTFDataFormatException (const char * *file*, const int *lineNumber*, const std::exception * *cause*, const char * *msg*, ...) throw () [inline]

Constructor - Initializes the file name and line number where this message occurred. Sets the message to report, using an optional list of arguments to parse into the message

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

cause The exception that was the cause for this one to be thrown.

msg The message to report

... list of primitives that are formatted into the message

6.582.2.5 decaf::io::UTFDataFormatException::UTFDataFormatException (const std::exception * *cause*) throw () [inline]

Constructor.

Parameters:

cause Pointer to the exception that caused this one to be thrown, the object is cloned caller retains ownership.

6.582.2.6 decaf::io::UTFDataFormatException::UTFDataFormatException (const char * *file*, const int *lineNumber*, const char * *msg*, ...) throw () [inline]

Constructor.

Parameters:

file The file name where exception occurs

lineNumber The line number where the exception occurred.

msg The message to report

... list of primitives that are formatted into the message

6.582.2.7 virtual decaf::io::UTFDataFormatException::~UTFDataFormatException () throw () [inline, virtual]

6.582.3 Member Function Documentation

6.582.3.1 virtual UTFDataFormatException* decaf::io::UTFDataFormatException::clone () const [inline, virtual]

Clones this exception. This is useful for cases where you need to preserve the type of the original exception as well as the message. All subclasses should override.

Returns:

A new instance of an Exception object that is a copy of this instance.

Reimplemented from **decaf::io::IOException** (p. 1479).

The documentation for this class was generated from the following file:

- src/main/decaf/io/**UTFDataFormatException.h**

6.583 decaf::util::UUID Class Reference

A class that represents an immutable universally unique identifier (**UUID** (p. 2686)).

`#include <src/main/decaf/util/UUID.h>`Inheritance diagram for decaf::util::UUID:

Public Member Functions

- **UUID** (long long mostSigBits, long long leastSigBits)

  *Constructs a new **UUID** (p. 2686) using the specified data.*

- virtual ~**UUID** ()
- virtual int **compareTo** (const **UUID** &value) const

  *Compare the given **UUID** (p. 2686) to this one.*

- virtual bool **equals** (const **UUID** &value) const

  *Compares this **UUID** (p. 2686) to the one given, returns true if they are equal.*

- virtual bool **operator==** (const **UUID** &value) const

  Compares equality between this object and the one passed.

- virtual bool **operator<** (const **UUID** &value) const

  Compares this object to another and returns true if this object is considered to be less than the one passed.

- virtual std::string **toString** () const

  *Returns a String object representing this **UUID** (p. 2686).*

- virtual long long **getLeastSignificantBits** () const
- virtual long long **getMostSignificantBits** () const
- virtual long long **node** () throw (lang::exceptions::UnsupportedOperationException)

  *The node value associated with this **UUID** (p. 2686).*

- virtual long long **timestamp** () throw (lang::exceptions::UnsupportedOperationException)

  *The timestamp value associated with this **UUID** (p. 2686).*

- virtual int **clockSequence** () throw (lang::exceptions::UnsupportedOperationException)

  *The clock sequence value associated with this **UUID** (p. 2686).*

- virtual int **variant** () throw (lang::exceptions::UnsupportedOperationException)

  *The variant number associated with this **UUID** (p. 2686).*

- virtual int **version** () throw (lang::exceptions::UnsupportedOperationException)

  *The version number associated with this **UUID** (p. 2686).*

Static Public Member Functions

- static **UUID randomUUID** ()

  Static factory to retrieve a type 4 (pseudo randomly generated) ***UUID*** *(p. 2686).*

- static **UUID nameUUIDFromBytes** (const std::vector< char > &name)

  Static factory to retrieve a type 3 (name based) ***UUID*** *(p. 2686) based on the specified byte array.*

- static **UUID nameUUIDFromBytes** (const char *name, std::size_t size)

  Static factory to retrieve a type 3 (name based) ***UUID*** *(p. 2686) based on the specified byte array.*

- static **UUID fromString** (const std::string &name) throw (lang::exceptions::IllegalArgumentException)

  Creates a ***UUID*** *(p. 2686) from the string standard representation as described in the* ***toString()*** *(p. 2691) method.*

6.583.1 Detailed Description

A class that represents an immutable universally unique identifier (**UUID** (p. 2686)). A **UUID** (p. 2686) represents a 128-bit value.

There exist different variants of these global identifiers. The methods of this class are for manipulating the Leach-Salz variant, although the constructors allow the creation of any variant of **UUID** (p. 2686) (described below).

The layout of a variant 2 (Leach-Salz) **UUID** (p. 2686) is as follows: The most significant long consists of the following unsigned fields:

0xFFFFFFFF00000000 time_low 0x00000000FFFF0000 time_mid 0x000000000000F000 version 0x0000000000000FFF time_hi

The least significant long consists of the following unsigned fields:

0xC000000000000000 variant 0x3FFF000000000000 clock_seq 0x0000FFFFFFFFFFFF node

The variant field contains a value which identifies the layout of the **UUID** (p. 2686). The bit layout described above is valid only for a **UUID** (p. 2686) with a variant value of 2, which indicates the Leach-Salz variant.

The version field holds a value that describes the type of this **UUID** (p. 2686). There are four different basic types of UUIDs: time-based, DCE **security** (p. 107), name-based, and randomly generated UUIDs. These types have a version value of 1, 2, 3 and 4, respectively.

For more information including algorithms used to create UUIDs, see the Internet-Draft UUIDs and GUIDs or the standards body definition at ISO/IEC 11578:1996.

6.583.2 Constructor & Destructor Documentation

6.583.2.1 decaf::util::UUID::UUID (long long *mostSigBits*, long long *leastSigBits*)

Constructs a new **UUID** (p. 2686) using the specified data. mostSigBits is used for the most significant 64 bits of the **UUID** (p. 2686) and leastSigBits becomes the least significant 64 bits of the **UUID** (p. 2686).

Parameters:

mostSigBits

leastSigBits

6.583.2.2 virtual decaf::util::UUID::~UUID () [virtual]

6.583.3 Member Function Documentation

6.583.3.1 virtual int decaf::util::UUID::clockSequence () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The clock sequence value associated with this **UUID** (p. 2686). The 14 bit clock sequence value is constructed from the clock sequence field of this **UUID** (p. 2686). The clock sequence field is used to guarantee temporal uniqueness in a time-based **UUID** (p. 2686).

The clockSequence value is only meaningful in a time-based **UUID** (p. 2686), which has version type 1. If this **UUID** (p. 2686) is not a time-based **UUID** (p. 2686) then this method throws UnsupportedOperationException.

Returns:

the clockSequeunce associated with a V1 **UUID** (p. 2686)

Exceptions:

UnsupportedOperationException

6.583.3.2 virtual int decaf::util::UUID::compareTo (const UUID & *value*) const [virtual]

Compare the given **UUID** (p. 2686) to this one.

Parameters:

value - the **UUID** (p. 2686) to compare to

6.583.3.3 virtual bool decaf::util::UUID::equals (const UUID & *value*) const [virtual]

Compares this **UUID** (p. 2686) to the one given, returns true if they are equal.

Parameters:

value - the **UUID** (p. 2686) to compare to.

Returns:

true if UUIDs are the same.

6.583.3.4 static UUID decaf::util::UUID::fromString (const std::string & *name*) throw (lang::exceptions::IllegalArgumentException) [static]

Creates a **UUID** (p. 2686) from the string standard representation as described in the **toString()** (p. 2691) method.

Parameters:

name - a string to be used to construct a **UUID** (p. 2686).

Returns:

type 3 **UUID** (p. 2686)

6.583.3.5 virtual long long decaf::util::UUID::getLeastSignificantBits () const [virtual]

Returns:

the most significant 64 bits of this UUID's 128 bit value.

6.583.3.6 virtual long long decaf::util::UUID::getMostSignificantBits () const [virtual]

Returns:

the most significant 64 bits of this UUID's 128 bit value.

6.583.3.7 static UUID decaf::util::UUID::nameUUIDFromBytes (const char * *name*, std::size_t *size*) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 2686) based on the specified byte array.

Parameters:

name - a byte array to be used to construct a **UUID** (p. 2686).

size - the size of the byte array, or number of bytes to use.

Returns:

type 3 **UUID** (p. 2686)

6.583.3.8 static UUID decaf::util::UUID::nameUUIDFromBytes (const std::vector< char > & *name*) [static]

Static factory to retrieve a type 3 (name based) **UUID** (p. 2686) based on the specified byte array.

Parameters:

name - a byte array to be used to construct a **UUID** (p. 2686).

Returns:

type 3 **UUID** (p. 2686)

6.583.3.9 virtual long long decaf::util::UUID::node () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The node value associated with this **UUID** (p. 2686). The 48 bit node value is constructed from the node field of this **UUID** (p. 2686). This field is intended to hold the IEEE 802 address of the machine that generated this **UUID** (p. 2686) to guarantee spatial uniqueness.

The node value is only meaningful in a time-based **UUID** (p. 2686), which has version type 1. If this **UUID** (p. 2686) is not a time-based **UUID** (p. 2686) then this method throws UnsupportedOperationException.

Returns:

the node value of this **UUID** (p. 2686)

Exceptions:

UnsupportedOperationException

6.583.3.10 virtual bool decaf::util::UUID::operator< (const UUID & *value*) const [virtual]

Compares this object to another and returns true if this object is considered to be less than the one passed. This

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.583.3.11 virtual bool decaf::util::UUID::operator== (const UUID & *value*) const [virtual]

Compares equality between this object and the one passed.

Parameters:

value - the value to be compared to this one.

Returns:

true if this object is equal to the one passed.

6.583.3.12 static UUID decaf::util::UUID::randomUUID () [static]

Static factory to retrieve a type 4 (pseudo randomly generated) **UUID** (p. 2686). The **UUID** (p. 2686) is generated using a cryptographically strong pseudo random number generator.

Returns:

type 4 **UUID** (p. 2686)

6.583.3.13 virtual long long decaf::util::UUID::timestamp () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The timestamp value associated with this **UUID** (p. 2686). The 60 bit timestamp value is constructed from the time_low, time_mid, and time_hi fields of this **UUID** (p. 2686). The resulting timestamp is measured in 100-nanosecond units since midnight, October 15, 1582 UTC.

The timestamp value is only meaningful in a time-based **UUID** (p. 2686), which has version type 1. If this **UUID** (p. 2686) is not a time-based **UUID** (p. 2686) then this method throws UnsupportedOperationException.

Returns:

the timestamp associated with a V1 **UUID** (p. 2686)

Exceptions:

UnsupportedOperationException

6.583.3.14 virtual std::string decaf::util::UUID::toString () const [virtual]

Returns a String object representing this **UUID** (p. 2686). UUID's are formatted as: 00112233-4455-6677-8899-AABBCCDDEEFF whose length is 36.

Returns:

formatted string for this **UUID** (p. 2686)

6.583.3.15 virtual int decaf::util::UUID::variant () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The variant number associated with this **UUID** (p. 2686). The variant number describes the layout of the **UUID** (p. 2686). The variant number has the following meaning:

∗ 0 Reserved for NCS backward compatibility ∗ 2 The Leach-Salz variant (used by this class) ∗ 6 Reserved, Microsoft Corporation backward compatibility ∗ 7 Reserved for future definition

Returns:

the variant associated with a V1 **UUID** (p. 2686)

Exceptions:

UnsupportedOperationException

6.583.3.16 virtual int decaf::util::UUID::version () throw (lang::exceptions::UnsupportedOperationException) [virtual]

The version number associated with this **UUID** (p. 2686). The version number describes how this **UUID** (p. 2686) was generated. The version number has the following meaning:

∗ 1 Time-based **UUID** (p. 2686) ∗ 2 DCE **security** (p. 107) **UUID** (p. 2686) ∗ 3 Name-based **UUID** (p. 2686) ∗ 4 Randomly generated **UUID** (p. 2686)

Returns:

the version associated with a V1 **UUID** (p. 2686)

Exceptions:

UnsupportedOperationException

The documentation for this class was generated from the following file:

- src/main/decaf/util/**UUID.h**

6.584 activemq::wireformat::WireFormat Class Reference

Provides a mechanism to marshal **commands** (p. 59) into and out of packets or into and out of streams, Channels and Datagrams.

`#include <src/main/activemq/wireformat/WireFormat.h>`Inheritance diagram for activemq::wireformat::WireFormat:

Public Member Functions

- virtual ~**WireFormat** ()
- virtual void **marshal** (const **Pointer**< **commands::Command** > &command, const **activemq::transport::Transport** *transport, **decaf::io::DataOutputStream** *out)=0 throw (decaf::io::IOException)

  Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

- virtual **Pointer**< **commands::Command** > **unmarshal** (const **activemq::transport::Transport** *transport, **decaf::io::DataInputStream** *in)=0 throw (decaf::io::IOException)

  Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form.

- virtual void **setVersion** (int version)=0

  Set the Version.

- virtual int **getVersion** () const =0

  Get the Version.

- virtual bool **hasNegotiator** () const =0

  *Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.*

- virtual **Pointer**< **transport::Transport** > **createNegotiator** (const **Pointer**< **transport::Transport** > &transport)=0 throw (decaf::lang::exceptions::UnsupportedOperationException)

  If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

6.584.1 Detailed Description

Provides a mechanism to marshal **commands** (p. 59) into and out of packets or into and out of streams, Channels and Datagrams.

Version:

Revision 1.1

6.584.2 Constructor & Destructor Documentation

6.584.2.1 virtual activemq::wireformat::WireFormat::~WireFormat () [inline, virtual]

6.584.3 Member Function Documentation

6.584.3.1 virtual Pointer<transport::Transport> activemq::wireformat::WireFormat::createNegotiator (const Pointer< transport::Transport > & *transport*) throw (decaf::lang::exceptions::UnsupportedOperationException) [pure virtual]

If the Transport Provides a Negotiator this method will create and return a new instance of the Negotiator.

Parameters:

transport *(p. 67)* - the Transport to Wrap the Negotiator around.

Returns:

new instance of a **WireFormatNegotiator** (p. 2721) as a **Pointer<Transport>** (p. 2011).

Exceptions:

UnsupportedOperationException if the **WireFormat** (p. 2693) doesn't have a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1973), and **activemq::wireformat::stomp::StompWireFormat** (p. 2467).

6.584.3.2 virtual int activemq::wireformat::WireFormat::getVersion () const [pure virtual]

Get the Version.

Returns:

the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1975), and **activemq::wireformat::stomp::StompWireFormat** (p. 2467).

6.584.3.3 virtual bool activemq::wireformat::WireFormat::hasNegotiator () const [pure virtual]

Returns true if this **WireFormat** (p. 2693) has a Negotiator that needs to wrap the Transport that uses it.

Returns:

true if the **WireFormat** (p. 2693) provides a Negotiator.

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1975), and **activemq::wireformat::stomp::StompWireFormat** (p. 2467).

6.584.3.4 virtual void activemq::wireformat::WireFormat::marshal (const Pointer< commands::Command > & *command*, const activemq::transport::Transport * *transport*, decaf::io::DataOutputStream * *out*) throw (decaf::io::IOException) [pure virtual]

Stream based marshaling of a Command, this method blocks until the entire Command has been written out to the output stream.

Parameters:

command The Command to Marshal

transport *(p. 67)* The Transport that called this method.

out The output stream to write the command to.

Exceptions:

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1977), and **activemq::wireformat::stomp::StompWireFormat** (p. 2468).

6.584.3.5 virtual void activemq::wireformat::WireFormat::setVersion (int *version*) [pure virtual]

Set the Version.

Parameters:

version the version of the wire format

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1979), and **activemq::wireformat::stomp::StompWireFormat** (p. 2468).

6.584.3.6 virtual Pointer<commands::Command> activemq::wireformat::WireFormat::unmarshal (const activemq::transport::Transport * *transport*, decaf::io::DataInputStream * *in*) throw (decaf::io::IOException) [pure virtual]

Stream based unmarshaling, blocks on reads on the input stream until a complete command has been read and unmarshaled into the correct form. Returns a Pointer to the newly unmarshaled Command.

Parameters:

transport *(p. 67)* - Pointer to the **transport** (p. 67) that is making this request.

in - the input stream to read the command from.

Returns:

the newly marshaled Command, caller owns the pointer

Exceptions:

IOException

Implemented in **activemq::wireformat::openwire::OpenWireFormat** (p. 1980), and **activemq::wireformat::stomp::StompWireFormat** (p. 2468).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormat.h**

6.585 activemq::wireformat::WireFormatFactory Class Reference

The **WireFormatFactory** (p. 2697) is the interface that all **WireFormatFactory** (p. 2697) classes must extend.

`#include <src/main/activemq/wireformat/WireFormatFactory.h>`Inheritance diagram for activemq::wireformat::WireFormatFactory:

Public Member Functions

- virtual ~**WireFormatFactory** ()
- virtual **Pointer**< **WireFormat** > **createWireFormat** (const **decaf::util::Properties** &properties)=0 throw (decaf::lang::exceptions::IllegalStateException)

  *Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.*

6.585.1 Detailed Description

The **WireFormatFactory** (p. 2697) is the interface that all **WireFormatFactory** (p. 2697) classes must extend. The Factory creates a **WireFormat** (p. 2693) Object based on the properties that are set in the passed `Properties` object.

6.585.2 Constructor & Destructor Documentation

6.585.2.1 virtual activemq::wireformat::WireFormatFactory::~WireFormatFactory () [inline, virtual]

6.585.3 Member Function Documentation

6.585.3.1 virtual Pointer<WireFormat> activemq::wireformat::WireFormatFactory::createWireFormat (const decaf::util::Properties & *properties*) throw (decaf::lang::exceptions::IllegalStateException) [pure virtual]

Creates a new **WireFormat** (p. 2693) Object passing it a set of properties from which it can obtain any optional settings.

Parameters:

properties - the Properties for this **WireFormat** (p. 2693)

Returns:

Pointer to a new instance of a **WireFormat** (p. 2693) object.

Implemented in **activemq::wireformat::openwire::OpenWireFormatFactory** (p. 1982), and **activemq::wireformat::stomp::StompWireFormatFactory** (p. 2470).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatFactory.h**

6.586 activemq::commands::WireFormatInfo Class Reference

`#include <src/main/activemq/commands/WireFormatInfo.h>`Inheritance diagram for activemq::commands::WireFormatInfo:

Public Member Functions

- **WireFormatInfo** ()
- virtual ~**WireFormatInfo** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **DataStructure** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this objects members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent.*

- virtual bool **isMarshalAware** () const

  Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

- virtual **decaf::lang::Pointer**< **commands::Command** > **visit** (**activemq::state::CommandVisitor** *visitor) throw (exceptions::ActiveMQException)

  Allows a Visitor to visit this command and return a response to the command based on the command type being visited.

- int **getVersion** () const

  Get the current Wireformat Version.

- void **setVersion** (int version)

  Set the current Wireformat Version.

- long long **getMaxInactivityDuration** () const

  Returns the currently configured Max Inactivity duration.

- void **setMaxInactivityDuration** (long long maxInactivityDuration)

  Sets the Max inactivity duration value.

- long long **getMaxInactivityDurationInitalDelay** () const

  Returns the currently configured Max Inactivity Intial Delay duration.

- void **setMaxInactivityDurationInitalDelay** (long long maxInactivityDurationInitalDelay)

  Sets the Max inactivity initial delay duration value.

- bool **isStackTraceEnabled** () const

  Checks if the stackTraceEnabled flag is on.

- void **setStackTraceEnabled** (bool stackTraceEnabled)

  Sets if the stackTraceEnabled flag is on.

- bool **isTcpNoDelayEnabled** () const

  Checks if the tcpNoDelayEnabled flag is on.

- void **setTcpNoDelayEnabled** (bool tcpNoDelayEnabled)

  Sets if the tcpNoDelayEnabled flag is on.

- bool **isCacheEnabled** () const

  Checks if the cacheEnabled flag is on.

- void **setCacheEnabled** (bool cacheEnabled)

  Sets if the cacheEnabled flag is on.

- int **getCacheSize** () const

  Gets the Cache Size setting.

- void **setCacheSize** (int value)

  Sets the Cache Size setting.

- bool **isTightEncodingEnabled** () const

  Checks if the tightEncodingEnabled flag is on.

- void **setTightEncodingEnabled** (bool tightEncodingEnabled)

  Sets if the tightEncodingEnabled flag is on.

- bool **isSizePrefixDisabled** () const

  Checks if the sizePrefixDisabled flag is on.

- void **setSizePrefixDisabled** (bool sizePrefixDisabled)

  Sets if the sizePrefixDisabled flag is on.

- const std::vector< unsigned char > & **getMagic** () const

  Get the Magic field.

- void **setMagic** (const std::vector< unsigned char > &magic)

  Sets the value of the magic field.

- const std::vector< unsigned char > & **getMarshalledProperties** () const

  Get the marshalledProperties field.

- void **setMarshalledProperties** (const std::vector< unsigned char > &marshalledProperties)

  Sets the value of the marshalledProperties field.

- virtual const **util::PrimitiveMap** & **getProperties** () const

  Gets the Properties for this ***Command*** *(p. 879).*

- virtual **util::PrimitiveMap** & **getProperties** ()

  Gets the Properties for this ***Command*** *(p. 879).*

- virtual void **setProperties** (const **util::PrimitiveMap** &map)

  Sets the Properties for this ***Command*** *(p. 879).*

- bool **isValid** () const

  Determines if we think this is a Valid ***WireFormatInfo*** *(p. 2699) command.*

- virtual bool **isWireFormatInfo** () const
- virtual void **beforeMarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

- virtual void **afterUnmarshal** (**wireformat::WireFormat** *wireFormat AMQCPP_UNUSED) throw (decaf::io::IOException)

  Called after unmarshaling is started to cleanup the object being unmarshaled.

Static Public Attributes

- static const unsigned char **ID_WIREFORMATINFO** = 1

6.586.1 Constructor & Destructor Documentation

6.586.1.1 activemq::commands::WireFormatInfo::WireFormatInfo ()

6.586.1.2 virtual activemq::commands::WireFormatInfo::~WireFormatInfo () [virtual]

6.586.2 Member Function Documentation

6.586.2.1 virtual void activemq::commands::WireFormatInfo::afterUnmarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [virtual]

Called after unmarshaling is started to cleanup the object being unmarshaled.

Parameters:

wireFormat - the **wireformat** (p. 74) object to control unmarshaling

Reimplemented from **activemq::commands::BaseDataStructure** (p. 558).

6.586.2.2 virtual void activemq::commands::WireFormatInfo::beforeMarshal (wireformat::WireFormat *wireFormat *AMQCPP_UNUSED*) throw (decaf::io::IOException) [virtual]

Handles the marshaling of the objects properties into the internal byte array before the object is marshalled to the wire.

Parameters:

wireFormat - the wire formatting controller

Reimplemented from **activemq::commands::BaseDataStructure** (p. 558).

6.586.2.3 virtual DataStructure* activemq::commands::WireFormatInfo::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Implements **activemq::commands::DataStructure** (p. 1174).

6.586.2.4 virtual void activemq::commands::WireFormatInfo::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this objects members, overwriting any existing data.

Returns:

src - Source Object

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.586.2.5 virtual bool activemq::commands::WireFormatInfo::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::BaseCommand** (p. 508).

6.586.2.6 int activemq::commands::WireFormatInfo::getCacheSize () const

Gets the Cache Size setting.

Returns:

currently set cache size.

6.586.2.7 virtual unsigned char activemq::commands::WireFormatInfo::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Implements **activemq::commands::DataStructure** (p. 1176).

6.586.2.8 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMagic () const [inline]

Get the Magic field.

Returns:

const reference to a std::vector<char>

6.586.2.9 const std::vector<unsigned char>& activemq::commands::WireFormatInfo::getMarshalledProperties () const [inline]

Get the marshalledProperties field.

Returns:

const reference to a std::vector<char>

6.586.2.10 long long activemq::commands::WireFormatInfo::getMaxInactivityDuration () const

Returns the currently configured Max Inactivity duration.

Returns:

the set inactivity duration value.

6.586.2.11 long long activemq::commands::WireFormatInfo::getMaxInactivityDurationInitalDelay () const

Returns the currently configured Max Inactivity Intial Delay duration.

Returns:

the set inactivity duration initial delay value.

6.586.2.12 virtual util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () [inline, virtual]

Gets the Properties for this **Command** (p. 879).

Returns:

the Properties object for this **Command** (p. 879).

6.586.2.13 virtual const util::PrimitiveMap& activemq::commands::WireFormatInfo::getProperties () const [inline, virtual]

Gets the Properties for this **Command** (p. 879).

Returns:

the Properties object for this **Command** (p. 879).

6.586.2.14 int activemq::commands::WireFormatInfo::getVersion () const [inline]

Get the current Wireformat Version.

Returns:

int that identifies the version

6.586.2.15 bool activemq::commands::WireFormatInfo::isCacheEnabled () const

Checks if the cacheEnabled flag is on.

Returns:

true if the flag is on.

6.586.2.16 virtual bool activemq::commands::WireFormatInfo::isMarshalAware () const [inline, virtual]

Indicates that this command is aware of Marshaling, and needs to have its Marshaling methods invoked.

Returns:

boolean indicating desire to be in marshaling stages

Reimplemented from **activemq::commands::BaseDataStructure** (p. 559).

6.586.2.17 bool activemq::commands::WireFormatInfo::isSizePrefixDisabled () const

Checks if the sizePrefixDisabled flag is on.

Returns:

true if the flag is on.

6.586.2.18 bool activemq::commands::WireFormatInfo::isStackTraceEnabled () const

Checks if the stackTraceEnabled flag is on.

Returns:

true if the flag is on.

6.586.2.19 bool activemq::commands::WireFormatInfo::isTcpNoDelayEnabled () const

Checks if the tcpNoDelayEnabled flag is on.

Returns:

true if the flag is on.

6.586.2.20 bool activemq::commands::WireFormatInfo::isTightEncodingEnabled () const

Checks if the tightEncodingEnabled flag is on.

Returns:

true if the flag is on.

6.586.2.21 bool activemq::commands::WireFormatInfo::isValid () const

Determines if we think this is a Valid **WireFormatInfo** (p. 2699) command.

Returns:

true if its valid.

6.586.2.22 virtual bool activemq::commands::WireFormatInfo::isWireFormatInfo () const [inline, virtual]

Returns:

answers true to the isWireFormatInfo query

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.586.2.23 void activemq::commands::WireFormatInfo::setCacheEnabled (bool *cacheEnabled*)

Sets if the cacheEnabled flag is on.

Parameters:

cacheEnabled - true to turn flag is on

6.586.2.24 void activemq::commands::WireFormatInfo::setCacheSize (int *value*)

Sets the Cache Size setting.

Parameters:

value - value to set to the cache size.

6.586.2.25 void activemq::commands::WireFormatInfo::setMagic (const std::vector< unsigned char > & *magic*) [inline]

Sets the value of the magic field.

Parameters:

magic - const std::vector<char>

6.586.2.26 void activemq::commands::WireFormatInfo::setMarshalledProperties (const std::vector< unsigned char > & *marshalledProperties*) [inline]

Sets the value of the marshalledProperties field.

Parameters:

marshalledProperties The Byte Array vector that contains the marshaled form of the **Message** (p. 1737) properties, this is the data sent over the wire.

6.586.2.27 void activemq::commands::WireFormatInfo::setMaxInactivityDuration (long long *maxInactivityDuration*)

Sets the Max inactivity duration value.

Parameters:

maxInactivityDuration - max time a client can be inactive.

6.586.2.28 void activemq::commands::WireFormatInfo::setMaxInactivityDurationInitalDelay (long long *maxInactivityDurationInitalDelay*)

Sets the Max inactivity initial delay duration value.

Parameters:

maxInactivityDurationInitalDelay - time before the inactivity delay is checked.

6.586.2.29 virtual void activemq::commands::WireFormatInfo::setProperties (const util::PrimitiveMap & *map*) [inline, virtual]

Sets the Properties for this **Command** (p. 879).

Parameters:

map - PrimtiveMap to copy

6.586.2.30 void activemq::commands::WireFormatInfo::setSizePrefixDisabled (bool *sizePrefixDisabled*)

Sets if the sizePrefixDisabled flag is on.

Parameters:

sizePrefixDisabled - true to turn flag is on

6.586.2.31 void activemq::commands::WireFormatInfo::setStackTraceEnabled (bool *stackTraceEnabled*)

Sets if the stackTraceEnabled flag is on.

Parameters:

stackTraceEnabled - ture to turn flag is on

6.586.2.32 void activemq::commands::WireFormatInfo::setTcpNoDelayEnabled (bool *tcpNoDelayEnabled*)

Sets if the tcpNoDelayEnabled flag is on.

Parameters:

tcpNoDelayEnabled - ture to turn flag is on

6.586.2.33 void activemq::commands::WireFormatInfo::setTightEncodingEnabled (bool *tightEncodingEnabled*)

Sets if the tightEncodingEnabled flag is on.

Parameters:

tightEncodingEnabled - true to turn flag is on

6.586.2.34 void activemq::commands::WireFormatInfo::setVersion (int *version*) [inline]

Set the current Wireformat Version.

Parameters:

version - int that identifies the version

6.586.2.35 virtual std::string activemq::commands::WireFormatInfo::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::BaseCommand** (p. 512).

6.586.2.36 virtual decaf::lang::Pointer<commands::Command> activemq::commands::WireFormatInfo::visit (activemq::state::CommandVisitor * *visitor*) throw (exceptions::ActiveMQException) [virtual]

Allows a Visitor to visit this command and return a response to the command based on the command type being visited. The command will call the proper processXXX method in the visitor.

Returns:

a **Response** (p. 2231) to the visitor being called or NULL if no response.

Implements **activemq::commands::Command** (p. 883).

6.586.3 Field Documentation

6.586.3.1 const unsigned char activemq::commands::WireFormatInfo::ID_WIREFORMATINFO = 1 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**WireFormatInfo.h**

6.587 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMar Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2709).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/WireFormatInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual ~**WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.587.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2709). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.587.2 Constructor & Destructor Documentation

6.587.2.1 activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::WireFormatInfo () [inline]

6.587.2.2 virtual activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::~WireFormatIn () [inline, virtual]

6.587.3 Member Function Documentation

6.587.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.587.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.587.3.3 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.587.3.4 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.587.3.5 virtual int activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.587.3.6 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.587.3.7 virtual void activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**WireFormatInfoMarshaller.h**

6.588 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMar Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2713).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/WireFormatInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual ~**WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.588.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2713). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.588.2 Constructor & Destructor Documentation

6.588.2.1 activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::WireFormatInfo () [inline]

6.588.2.2 virtual activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::~WireFormatIn () [inline, virtual]

6.588.3 Member Function Documentation

6.588.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.588.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.588.3.3 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.588.3.4 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.588.3.5 virtual int activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.588.3.6 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.588.3.7 virtual void activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**WireFormatInfoMarshaller.h**

6.589 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMar Class Reference

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2717).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/WireFormatInfoMarshaller.h>`Inheritance diagram for activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller:

Public Member Functions

- **WireFormatInfoMarshaller** ()
- virtual ~**WireFormatInfoMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.589.1 Detailed Description

Marshaling code for Open Wire Format for **WireFormatInfoMarshaller** (p. 2717). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.589.2 Constructor & Destructor Documentation

6.589.2.1 activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::WireFormatInfo () [inline]

6.589.2.2 virtual activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::~WireFormatIn () [inline, virtual]

6.589.3 Member Function Documentation

6.589.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.589.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::getDataStructureT () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.589.3.3 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1154).

6.589.3.4 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1158).

6.589.3.5 virtual int activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1162).

6.589.3.6 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1166).

6.589.3.7 virtual void activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1169).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**WireFormatInfoMarshaller.h**

6.590 activemq::wireformat::WireFormatNegotiator Class Reference

Defines a **WireFormatNegotiator** (p. 2721) which allows a **WireFormat** (p. 2693) to.

`#include <src/main/activemq/wireformat/WireFormatNegotiator.h>`Inheritance diagram for activemq::wireformat::WireFormatNegotiator:

Public Member Functions

- **WireFormatNegotiator** (const **Pointer**< **transport::Transport** > &**next**)
  Constructor.
- virtual ~**WireFormatNegotiator** ()

6.590.1 Detailed Description

Defines a **WireFormatNegotiator** (p. 2721) which allows a **WireFormat** (p. 2693) to.

6.590.2 Constructor & Destructor Documentation

6.590.2.1 activemq::wireformat::WireFormatNegotiator::WireFormatNegotiator (const Pointer< transport::Transport > & *next*) [inline]

Constructor.

Parameters:

next - the next Transport in the chain

6.590.2.2 virtual activemq::wireformat::WireFormatNegotiator::~WireFormatNegotiator () [inline, virtual]

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatNegotiator.h**

6.591 activemq::wireformat::WireFormatRegistry Class Reference

Registry of all **WireFormat** (p. 2693) Factories that are available to the client at runtime.

`#include <src/main/activemq/wireformat/WireFormatRegistry.h>`

Public Member Functions

- virtual ~**WireFormatRegistry** ()
- **WireFormatFactory** * **findFactory** (const std::string &name) const throw (decaf::lang::exceptions::NoSuchElementException)

  *Gets a Registered **WireFormatFactory** (p. 2697) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.*

- void **registerFactory** (const std::string &name, **WireFormatFactory** *factory) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

  *Registers a new **WireFormatFactory** (p. 2697) with this Registry.*

- void **unregisterFactory** (const std::string &name)

  Unregisters the Factory with the given name and deletes that instance of the Factory.

- std::vector< std::string > **getWireFormatNames** () const

  Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Static Public Member Functions

- static **WireFormatRegistry** & **getInstance** ()

  *Gets the single instance of the **WireFormatRegistry** (p. 2722).*

6.591.1 Detailed Description

Registry of all **WireFormat** (p. 2693) Factories that are available to the client at runtime. New WireFormat's must have a factory registered here before a connection attempt is made.

Since:

3.0

6.591.2 Constructor & Destructor Documentation

6.591.2.1 virtual activemq::wireformat::WireFormatRegistry::~WireFormatRegistry () [virtual]

6.591.3 Member Function Documentation

6.591.3.1 WireFormatFactory* activemq::wireformat::WireFormatRegistry::findFactory (const std::string & *name*) const throw (decaf::lang::exceptions::NoSuchElementException)

Gets a Registered **WireFormatFactory** (p. 2697) from the Registry and returns it if there is not a registered format factory with the given name an exception is thrown.

Parameters:

name The name of the Factory to find in the Registry.

Returns:

the Factory registered under the given name.

Exceptions:

NoSuchElementException if no factory is registered with that name.

6.591.3.2 static WireFormatRegistry& activemq::wireformat::WireFormatRegistry::getInstance () [static]

Gets the single instance of the **WireFormatRegistry** (p. 2722).

Returns:

reference to the single instance of this Registry

6.591.3.3 std::vector<std::string> activemq::wireformat::WireFormatRegistry::getWireFormatNames () const

Retrieves a list of the names of all the Registered WireFormat's in this Registry.

Returns:

stl vector of strings with all the **WireFormat** (p. 2693) names registered.

6.591.3.4 void activemq::wireformat::WireFormatRegistry::registerFactory (const std::string & *name*, WireFormatFactory * *factory*) throw (decaf::lang::exceptions::IllegalArgumentException, decaf::lang::exceptions::NullPointerException)

Registers a new **WireFormatFactory** (p. 2697) with this Registry. If a Factory with the given name is already registered it is overwritten with the new one. Once a factory is added to the Registry its lifetime is controlled by the Registry, it will be deleted once the Registry has been deleted.

Parameters:

name The name of the new Factory to register.

factory The new Factory to add to the Registry.

Exceptions:

IllegalArgumentException is name is the empty string.

NullPointerException if the Factory is Null.

6.591.3.5 void activemq::wireformat::WireFormatRegistry::unregisterFactory (const std::string & *name*)

Unregisters the Factory with the given name and deletes that instance of the Factory.

Parameters:

name Name of the Factory to unregister and destroy

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/**WireFormatRegistry.h**

6.592 decaf::io::Writer Class Reference

`#include <src/main/decaf/io/Writer.h>`

Public Member Functions

- virtual ~**Writer** ()
- virtual void **setOutputStream** (**OutputStream** *os)=0

  Sets the target output stream.

- virtual **OutputStream** * **getOutputStream** ()=0

  Gets the target output stream.

- virtual void **write** (const unsigned char *buffer, std::size_t count)=0 throw (IOException, lang::exceptions::NullPointerException)

  Writes a byte array to the output stream.

- virtual void **writeByte** (unsigned char v)=0 throw (IOException)

  Writes a byte to the output stream.

6.592.1 Constructor & Destructor Documentation

6.592.1.1 virtual decaf::io::Writer::~Writer () [inline, virtual]

6.592.2 Member Function Documentation

6.592.2.1 virtual OutputStream* decaf::io::Writer::getOutputStream () [pure virtual]

Gets the target output stream.

Returns:

the output stream currently being used

6.592.2.2 virtual void decaf::io::Writer::setOutputStream (OutputStream * *os*) [pure virtual]

Sets the target output stream.

Parameters:

os The provided Outputstream to use to write to.

6.592.2.3 virtual void decaf::io::Writer::write (const unsigned char * *buffer*, std::size_t *count*) throw (IOException, lang::exceptions::NullPointerException) [pure virtual]

Writes a byte array to the output stream.

Parameters:

buffer a byte array

count the number of bytes in the array to write.

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

6.592.2.4 virtual void decaf::io::Writer::writeByte (unsigned char *v*) throw (IOException) [pure virtual]

Writes a byte to the output stream.

Parameters:

v The value to be written.

Exceptions:

IOException *(*p. *1477)* thrown if an error occurs.

The documentation for this class was generated from the following file:

- src/main/decaf/io/**Writer.h**

6.593 decaf::security::auth::x500::X500Principal Class Reference

`#include <src/main/decaf/security/auth/x500/X500Principal.h>`Inheritance diagram for decaf::security::auth::x500::X500Principal:

Public Member Functions

- virtual ~**X500Principal** ()
- virtual std::string **getName** () const =0

  Provides the name of this principal.

- virtual void **getEncoded** (std::vector< unsigned char > &output) const =0
- virtual int **hashCode** () const =0

6.593.1 Constructor & Destructor Documentation

6.593.1.1 virtual decaf::security::auth::x500::X500Principal::~X500Principal () [inline, virtual]

6.593.2 Member Function Documentation

6.593.2.1 virtual void decaf::security::auth::x500::X500Principal::getEncoded (std::vector< unsigned char > & *output*) const [pure virtual]

6.593.2.2 virtual std::string decaf::security::auth::x500::X500Principal::getName () const [pure virtual]

Provides the name of this principal.

Returns:

the name of this principal.

Implements **decaf::security::Principal** (p. 2084).

6.593.2.3 virtual int decaf::security::auth::x500::X500Principal::hashCode () const [pure virtual]

The documentation for this class was generated from the following file:

- src/main/decaf/security/auth/x500/**X500Principal.h**

6.594 decaf::security::cert::X509Certificate Class Reference

Base interface for all identity certificates.

`#include <src/main/decaf/security/cert/X509Certificate.h>`Inheritance diagram for decaf::security::cert::X509Certificate:

Public Member Functions

- virtual ~**X509Certificate** ()
- virtual void **checkValidity** () const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual void **checkValidity** (const **decaf::util::Date** &date) const =0 throw (CertificateExpiredException, CertificateNotYetValidException)
- virtual int **getBasicConstraints** () const =0
- virtual void **getIssuerUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getIssuerX500Principal** () const =0
- virtual void **getKeyUsage** (std::vector< unsigned char > &output) const =0
- virtual Date **getNotAfter** () const =0
- virtual Date **getNotBefore** () const =0
- virtual std::string **getSigAlgName** () const =0
- virtual std::string **getSigAlgOID** () const =0
- virtual void **getSigAlgParams** (std::vector< unsigned char > &output) const =0
- virtual void **getSignature** (std::vector< unsigned char > &output) const =0
- virtual void **getSubjectUniqueID** (std::vector< bool > &output) const =0
- virtual const X500Principal * **getSubjectX500Principal** () const =0
- virtual void **getTBSCertificate** (std::vector< unsigned char > &output) const =0 throw (CertificateEncodingException)
- virtual int **getVersion** () const =0

6.594.1 Detailed Description

Base interface for all identity certificates.

6.594.2 Constructor & Destructor Documentation

6.594.2.1 virtual decaf::security::cert::X509Certificate::~X509Certificate () [inline, virtual]

6.594.3 Member Function Documentation

6.594.3.1 virtual void decaf::security::cert::X509Certificate::checkValidity (const decaf::util::Date & *date*) const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1965).

6.594.3.2 virtual void decaf::security::cert::X509Certificate::checkValidity () const throw (CertificateExpiredException, CertificateNotYetValidException) [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1965).

6.594.3.3 virtual int decaf::security::cert::X509Certificate::getBasicConstraints () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.4 virtual void decaf::security::cert::X509Certificate::getIssuerUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.5 virtual const X500Principal* decaf::security::cert::X509Certificate::getIssuerX500Principal () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.6 virtual void decaf::security::cert::X509Certificate::getKeyUsage (std::vector< unsigned char > & *output*) const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.7 virtual Date decaf::security::cert::X509Certificate::getNotAfter () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.8 virtual Date decaf::security::cert::X509Certificate::getNotBefore () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1966).

6.594.3.9 virtual std::string decaf::security::cert::X509Certificate::getSigAlgName () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.594.3.10 virtual std::string decaf::security::cert::X509Certificate::getSigAlgOID () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.594.3.11 virtual void decaf::security::cert::X509Certificate::getSigAlgParams (std::vector< unsigned char > & *output*) const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.594.3.12 virtual void decaf::security::cert::X509Certificate::getSignature (std::vector< unsigned char > & *output*) const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1967).

6.594.3.13 virtual void decaf::security::cert::X509Certificate::getSubjectUniqueID (std::vector< bool > & *output*) const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

6.594.3.14 virtual const X500Principal* decaf::security::cert::X509Certificate::getSubjectX500Principal () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

6.594.3.15 virtual void decaf::security::cert::X509Certificate::getTBSCertificate (std::vector< unsigned char > & *output*) const throw (CertificateEncodingException) [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

6.594.3.16 virtual int decaf::security::cert::X509Certificate::getVersion () const [pure virtual]

Implemented in **decaf::security_provider::unix::openssl::OpenSSLX509Certificate** (p. 1968).

The documentation for this class was generated from the following file:

- src/main/decaf/security/cert/**X509Certificate.h**

6.595 activemq::commands::XATransactionId Class Reference

`#include <src/main/activemq/commands/XATransactionId.h>`Inheritance diagram for activemq::commands::XATransactionId:

Public Types

- typedef **decaf::lang::PointerComparator< XATransactionId > COMPARATOR**

Public Member Functions

- **XATransactionId** ()
- **XATransactionId** (const **XATransactionId** &other)
- virtual ~**XATransactionId** ()
- virtual unsigned char **getDataStructureType** () const

  Get the unique identifier that this object and its own Marshaler share.

- virtual **XATransactionId** * **cloneDataStructure** () const

  Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

- virtual void **copyDataStructure** (const **DataStructure** *src)

  Copy the contents of the passed object into this object's members, overwriting any existing data.

- virtual std::string **toString** () const

  *Returns a string containing the information for this **DataStructure*** (p. *1174*) *such as its type and value of its elements.*

- virtual bool **equals** (const **DataStructure** *value) const

  *Compares the **DataStructure*** (p. *1174*) *passed in to this one, and returns if they are equivalent.*

- virtual int **getFormatId** () const
- virtual void **setFormatId** (int **formatId**)
- virtual const std::vector< unsigned char > & **getGlobalTransactionId** () const
- virtual std::vector< unsigned char > & **getGlobalTransactionId** ()
- virtual void **setGlobalTransactionId** (const std::vector< unsigned char > &**globalTransactionId**)
- virtual const std::vector< unsigned char > & **getBranchQualifier** () const
- virtual std::vector< unsigned char > & **getBranchQualifier** ()
- virtual void **setBranchQualifier** (const std::vector< unsigned char > &**branchQualifier**)
- virtual int **compareTo** (const **XATransactionId** &value) const
- virtual bool **equals** (const **XATransactionId** &value) const
- virtual bool **operator==** (const **XATransactionId** &value) const
- virtual bool **operator<** (const **XATransactionId** &value) const
- **XATransactionId** & **operator=** (const **XATransactionId** &other)

Static Public Attributes

- static const unsigned char **ID_XATRANSACTIONID** = 112

Protected Attributes

- int **formatId**
- std::vector< unsigned char > **globalTransactionId**
- std::vector< unsigned char > **branchQualifier**

6.595.1 Member Typedef Documentation

6.595.1.1 typedef decaf::lang::PointerComparator<XATransactionId> activemq::commands::XATransactionId::COMPARATOR

Reimplemented from **activemq::commands::TransactionId** (p. 2573).

6.595.2 Constructor & Destructor Documentation

6.595.2.1 activemq::commands::XATransactionId::XATransactionId ()

6.595.2.2 activemq::commands::XATransactionId::XATransactionId (const XATransactionId & *other*)

6.595.2.3 virtual activemq::commands::XATransactionId::~XATransactionId () [virtual]

6.595.3 Member Function Documentation

6.595.3.1 virtual XATransactionId* activemq::commands::XATransactionId::cloneDataStructure () const [virtual]

Clone this object and return a new instance that the caller now owns, this will be an exact copy of this one.

Returns:

new copy of this object.

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.595.3.2 virtual int activemq::commands::XATransactionId::compareTo (const XATransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.595.3.3 virtual void activemq::commands::XATransactionId::copyDataStructure (const DataStructure * *src*) [virtual]

Copy the contents of the passed object into this object's members, overwriting any existing data.

Parameters:

src - Source Object

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.595.3.4 virtual bool activemq::commands::XATransactionId::equals (const XATransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2574).

6.595.3.5 virtual bool activemq::commands::XATransactionId::equals (const DataStructure * *value*) const [virtual]

Compares the **DataStructure** (p. 1174) passed in to this one, and returns if they are equivalent. Equivalent here means that they are of the same type, and that each element of the objects are the same.

Returns:

true if DataStructure's are Equal.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.3.6 virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () [virtual]

6.595.3.7 virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getBranchQualifier () const [virtual]

6.595.3.8 virtual unsigned char activemq::commands::XATransactionId::getDataStructureType () const [virtual]

Get the unique identifier that this object and its own Marshaler share.

Returns:

new **DataStructure** (p. 1174) type copy.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.3.9 virtual int activemq::commands::XATransactionId::getFormatId () const [virtual]

6.595.3.10 virtual std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId () [virtual]

6.595.3.11 virtual const std::vector<unsigned char>& activemq::commands::XATransactionId::getGlobalTransactionId () const [virtual]

6.595.3.12 virtual bool activemq::commands::XATransactionId::operator< (const XATransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.3.13 XATransactionId& activemq::commands::XATransactionId::operator= (const XATransactionId & *other*)

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.3.14 virtual bool activemq::commands::XATransactionId::operator== (const XATransactionId & *value*) const [virtual]

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.3.15 virtual void activemq::commands::XATransactionId::setBranchQualifier (const std::vector< unsigned char > & *branchQualifier*) [virtual]

6.595.3.16 virtual void activemq::commands::XATransactionId::setFormatId (int *formatId*) [virtual]

6.595.3.17 virtual void activemq::commands::XATransactionId::setGlobalTransactionId (const std::vector< unsigned char > & *globalTransactionId*) [virtual]

6.595.3.18 virtual std::string activemq::commands::XATransactionId::toString () const [virtual]

Returns a string containing the information for this **DataStructure** (p. 1174) such as its type and value of its elements.

Returns:

formatted string useful for debugging.

Reimplemented from **activemq::commands::TransactionId** (p. 2575).

6.595.4 Field Documentation

6.595.4.1 std::vector<unsigned char> activemq::commands::XATransactionId::branchQualifier [protected]

6.595.4.2 int activemq::commands::XATransactionId::formatId [protected]

6.595.4.3 std::vector<unsigned char> activemq::commands::XATransactionId::globalTransactionId [protected]

6.595.4.4 const unsigned char activemq::commands::XATransactionId::ID_XATRANSACTIONID = 112 [static]

The documentation for this class was generated from the following file:

- src/main/activemq/commands/**XATransactionId.h**

6.596 activemq::wireformat::openwire::marshal::v1::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2736).

`#include <src/main/activemq/wireformat/openwire/marshal/v1/XATransactionIdMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual ~**XATransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.596.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2736). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.596.2 Constructor & Destructor Documentation

6.596.2.1 activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::XATransaction () [inline]

6.596.2.2 virtual activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::~XATransactio () [inline, virtual]

6.596.3 Member Function Documentation

6.596.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.596.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.596.3.3 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2578).

6.596.3.4 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2578).

6.596.3.5 virtual int activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2579).

6.596.3.6 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2579).

6.596.3.7 virtual void activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller** (p. 2580).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v1/**XATransactionIdMarshaller.h**

6.597 activemq::wireformat::openwire::marshal::v3::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2740).

`#include <src/main/activemq/wireformat/openwire/marshal/v3/XATransactionIdMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual ~**XATransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.597.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2740). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.597.2 Constructor & Destructor Documentation

6.597.2.1 activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::XATransaction () [inline]

6.597.2.2 virtual activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::~XATransactio () [inline, virtual]

6.597.3 Member Function Documentation

6.597.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::createObject () const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.597.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::getDataStructure () const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.597.3.3 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseMarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582).

6.597.3.4 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2582).

6.597.3.5 virtual int activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2583).

6.597.3.6 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2583).

6.597.3.7 virtual void activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller** (p. 2584).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v3/**XATransactionIdMarshaller.h**

6.598 activemq::wireformat::openwire::marshal::v2::XATransactionIdMa Class Reference

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2744).

`#include <src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h>`Inheritanc diagram for activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller:

Public Member Functions

- **XATransactionIdMarshaller** ()
- virtual ~**XATransactionIdMarshaller** ()
- virtual **commands::DataStructure** * **createObject** () const

  Creates a new instance of this marshalable type.

- virtual unsigned char **getDataStructureType** () const

  Get the Data Structure Type that identifies this Marshaler.

- virtual void **tightUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual int **tightMarshal1** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write the booleans that this object uses to a BooleanStream.

- virtual void **tightMarshal2** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut, **utils::BooleanStream** *bs) throw (decaf::io::IOException)

  Write a object instance to data output stream.

- virtual void **looseUnmarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataInputStream** *dataIn) throw (decaf::io::IOException)

  Un-marshal an object instance from the data input stream.

- virtual void **looseMarshal** (**OpenWireFormat** *wireFormat, **commands::DataStructure** *dataStructure, **decaf::io::DataOutputStream** *dataOut) throw (decaf::io::IOException)

  Write a object instance to data output stream.

6.598.1 Detailed Description

Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2744). NOTE!: This file is auto generated - do not modify! if you need to make a change, please see the Java Classes in the activemq-openwire-generator module

6.598.2 Constructor & Destructor Documentation

6.598.2.1 activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::XATransaction() [inline]

6.598.2.2 virtual activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::~XATransactio() [inline, virtual]

6.598.3 Member Function Documentation

6.598.3.1 virtual commands::DataStructure* activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::createObject() const [virtual]

Creates a new instance of this marshalable type.

Returns:

new DataStructure object pointer caller owns it.

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1148).

6.598.3.2 virtual unsigned char activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::getDataStructure() const [virtual]

Get the Data Structure Type that identifies this Marshaler.

Returns:

byte holding the data structure type value

Implements **activemq::wireformat::openwire::marshal::DataStreamMarshaller** (p. 1151).

6.598.3.3 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseMarshal(OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describs the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryWriter that provides that data sink

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2586).

6.598.3.4 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::looseUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataIn - BinaryReader that provides that data source

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2586).

6.598.3.5 virtual int activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal1 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write the booleans that this object uses to a BooleanStream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

bs - BooleanStream stream used to pack bits from the wire.

Returns:

int value indicating the size of the marshaled object.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2587).

6.598.3.6 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightMarshal2 (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataOutputStream * *dataOut*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Write a object instance to data output stream.

Parameters:

wireFormat - describes the wire format of the broker

dataStructure - Object to be marshaled

dataOut - BinaryReader that provides that data sink

bs - BooleanStream stream used to pack bits from the wire.

Exceptions:

IOException if an error occurs during the **marshal** (p. 76).

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2587).

6.598.3.7 virtual void activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller::tightUnmarshal (OpenWireFormat * *wireFormat*, commands::DataStructure * *dataStructure*, decaf::io::DataInputStream * *dataIn*, utils::BooleanStream * *bs*) throw (decaf::io::IOException) [virtual]

Un-marshal an object instance from the data input stream.

Parameters:

wireFormat - describes the wire format of the broker.

dataStructure - Object to be un-marshaled.

dataIn - BinaryReader that provides that data.

bs - BooleanStream stream used to unpack bits from the wire.

Exceptions:

IOException if an error occurs during the unmarshal.

Reimplemented from **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller** (p. 2588).

The documentation for this class was generated from the following file:

- src/main/activemq/wireformat/openwire/marshal/v2/**XATransactionIdMarshaller.h**

Chapter 7

File Documentation

7.1 src/main/activemq/cmsutil/CachedConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::CachedConsumer**

  A cached message consumer contained within a pooled session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.2 src/main/activemq/cmsutil/CachedProducer.h File Reference

`#include <cms/MessageProducer.h>`

`#include <activemq/util/Config.h>`

Data Structures

- class **activemq::cmsutil::CachedProducer**

  A cached message producer contained within a pooled session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.3 src/main/activemq/cmsutil/CmsAccessor.h File Reference

```
#include <cms/ConnectionFactory.h>
#include <activemq/cmsutil/ResourceLifecycleManager.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::cmsutil::CmsAccessor**

  *Base class for **activemq.cmsutil.CmsTemplate** (p. 858) and other CMS-accessing gateway helpers, defining common properties such as the CMS **cms.ConnectionFactory** (p. 978) to operate on.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.4 src/main/activemq/cmsutil/CmsDestinationAccessor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsAccessor.h>
#include <activemq/cmsutil/DynamicDestinationResolver.h>
```

Data Structures

- class **activemq::cmsutil::CmsDestinationAccessor**

  Extends the `CmsAccessor` *(* p. *843) to add support for resolving destination names.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.5 src/main/activemq/cmsutil/CmsTemplate.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/cmsutil/CmsDestinationAccessor.h>
#include <activemq/cmsutil/SessionCallback.h>
#include <activemq/cmsutil/ProducerCallback.h>
#include <activemq/cmsutil/SessionPool.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <cms/ConnectionFactory.h>
#include <cms/DeliveryMode.h>
#include <string>
```

Data Structures

- class **activemq::cmsutil::CmsTemplate**

  CmsTemplate (p. 858) simplifies performing synchronous CMS operations.

- class **activemq::cmsutil::CmsTemplate::ProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveProducerExecutor**
- class **activemq::cmsutil::CmsTemplate::SendExecutor**
- class **activemq::cmsutil::CmsTemplate::ReceiveExecutor**
- class **activemq::cmsutil::CmsTemplate::ResolveReceiveExecutor**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.6 src/main/activemq/cmsutil/DestinationResolver.h File Reference

`#include <cms/Session.h>`

`#include <activemq/util/Config.h>`

Data Structures

- class **activemq::cmsutil::DestinationResolver**

  Resolves a CMS destination name to a ***Destination****.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.7 src/main/activemq/cmsutil/DynamicDestinationResolver.h File Reference

```
#include <activemq/cmsutil/DestinationResolver.h>
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::DynamicDestinationResolver**

  Resolves a CMS destination name to a `Destination`*.*

- class **activemq::cmsutil::DynamicDestinationResolver::SessionResolver**

  Manages maps of names to topics and queues for a single session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.8 src/main/activemq/cmsutil/MessageCreator.h File Reference

```
#include <cms/Session.h>
#include <cms/Message.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::MessageCreator**

  Creates the user-defined message to be sent by the `CmsTemplate` (p. *858*).

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.9 src/main/activemq/cmsutil/PooledSession.h File Reference

```
#include <cms/Session.h>
#include <decaf/util/StlMap.h>
#include <activemq/cmsutil/CachedProducer.h>
#include <activemq/cmsutil/CachedConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::PooledSession**

  A pooled session object that wraps around a delegate session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.10 src/main/activemq/cmsutil/ProducerCallback.h File Reference

```
#include <cms/Session.h>
#include <cms/MessageProducer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ProducerCallback**

  Callback for sending a message to a CMS destination.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.11 src/main/activemq/cmsutil/ResourceLifecycleManager.h File Reference

```
#include <cms/Connection.h>
#include <cms/Session.h>
#include <cms/Destination.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::ResourceLifecycleManager**

  Manages the lifecycle of a set of CMS resources.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.12 src/main/activemq/cmsutil/SessionCallback.h File Reference

`#include <cms/Session.h>`

`#include <activemq/util/Config.h>`

Data Structures

- class **activemq::cmsutil::SessionCallback**

  Callback for executing any number of operations on a provided CMS Session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.13 src/main/activemq/cmsutil/SessionPool.h File Reference

```
#include <activemq/cmsutil/PooledSession.h>
#include <decaf/util/concurrent/Mutex.h>
#include <cms/Connection.h>
#include <list>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::cmsutil::SessionPool**

  A pool of CMS sessions from the same connection and with the same acknowledge mode.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::cmsutil**

7.14 src/main/activemq/commands/ActiveMQBlobMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/Message.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBlobMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.15 src/main/activemq/commands/ActiveMQBytesMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <decaf/io/ByteArrayInputStream.h>
#include <decaf/io/ByteArrayOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <cms/BytesMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQBytesMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.16 src/main/activemq/commands/ActiveMQDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/ActiveMQProperties.h>
#include <cms/Destination.h>
#include <decaf/lang/Pointer.h>
#include <vector>
#include <string>
#include <map>
```

Data Structures

- class **activemq::commands::ActiveMQDestination**
- struct **activemq::commands::ActiveMQDestination::DestinationFilter**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.17 src/main/activemq/commands/ActiveMQMapMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <cms/MapMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQMapMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.18 src/main/activemq/commands/ActiveMQMessage.h File Reference

```
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.19 src/main/activemq/commands/ActiveMQMessageTemplate.h File Reference

```
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/wireformat/openwire/utils/MessagePropertyInterceptor.h>
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
```

Data Structures

- class **activemq::commands::ActiveMQMessageTemplate< T >**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.20 src/main/activemq/commands/ActiveMQObjectMessage.h File Reference

```
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/ObjectMessage.h>
#include <activemq/util/Config.h>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQObjectMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.21 src/main/activemq/commands/ActiveMQQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Queue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQQueue**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.22 src/main/activemq/commands/ActiveMQStreamMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveList.h>
#include <activemq/commands/ActiveMQMessage.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/StreamMessage.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQStreamMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.23 src/main/activemq/commands/ActiveMQTempDestination.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <cms/Closeable.h>
#include <vector>
#include <string>
```

Data Structures

- class **activemq::commands::ActiveMQTempDestination**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.24 src/main/activemq/commands/ActiveMQTempQueue.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryQueue.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempQueue**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.25 src/main/activemq/commands/ActiveMQTempTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <cms/TemporaryTopic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTempTopic**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.26 src/main/activemq/commands/ActiveMQTextMessage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQMessageTemplate.h>
#include <cms/TextMessage.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTextMessage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.27 src/main/activemq/commands/ActiveMQTopic.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Exception.h>
#include <cms/Topic.h>
#include <vector>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::commands::ActiveMQTopic**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.28 src/main/activemq/commands/BaseCommand.h File Reference

`#include <activemq/util/Config.h>`

`#include <activemq/commands/Command.h>`

Data Structures

- class **activemq::commands::BaseCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.29 src/main/activemq/commands/BaseDataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <string>
#include <sstream>
```

Data Structures

- class **activemq::commands::BaseDataStructure**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::commands**

7.30 src/main/activemq/commands/BooleanExpression.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::commands::BooleanExpression**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.31 src/main/activemq/commands/BrokerError.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerError**

  This class represents an Exception sent from the Broker.

- struct **activemq::commands::BrokerError::StackTraceElement**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.32 src/main/activemq/commands/BrokerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.33 src/main/activemq/commands/BrokerInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::BrokerInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.34 src/main/activemq/commands/Command.h File Reference

```
#include <string>
#include <activemq/util/Config.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::commands::Command**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**
- namespace **activemq::commands**

7.35 src/main/activemq/commands/ConnectionControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionControl**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.36 src/main/activemq/commands/ConnectionError.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionError**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.37 src/main/activemq/commands/ConnectionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.38 src/main/activemq/commands/ConnectionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConnectionInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.39 src/main/activemq/commands/ConsumerControl.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerControl**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.40 src/main/activemq/commands/ConsumerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.41 src/main/activemq/commands/ConsumerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BooleanExpression.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ConsumerInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.42 src/main/activemq/commands/ControlCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ControlCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.43 src/main/activemq/commands/DataArrayResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DataArrayResponse**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.44 src/main/activemq/commands/DataResponse.h File Reference

```
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DataResponse**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.45 src/main/activemq/commands/DataStructure.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/MarshalAware.h>
```

Data Structures

- class **activemq::commands::DataStructure**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.46 src/main/activemq/commands/DestinationInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DestinationInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.47 src/main/activemq/commands/DiscoveryEvent.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::DiscoveryEvent**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.48 src/main/activemq/commands/ExceptionResponse.h File Reference

```
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ExceptionResponse**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.49 src/main/activemq/commands/FlushCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::FlushCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.50 src/main/activemq/commands/IntegerResponse.h File Reference

```
#include <activemq/commands/Response.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::IntegerResponse**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.51 src/main/activemq/commands/JournalQueueAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalQueueAck**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.52 src/main/activemq/commands/JournalTopicAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTopicAck**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.53 src/main/activemq/commands/JournalTrace.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTrace**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.54 src/main/activemq/commands/JournalTransaction.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::JournalTransaction**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.55 src/main/activemq/commands/KeepAliveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::KeepAliveInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.56 src/main/activemq/commands/LastPartialCommand.h File Reference

```
#include <activemq/commands/PartialCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::LastPartialCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.57 src/main/activemq/commands/LocalTransactionId.h File Reference

```
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::LocalTransactionId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.58 src/main/activemq/commands/Message.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveMap.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Date.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Message**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**
- namespace **activemq::commands**

7.59 src/main/cms/Message.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
#include <cms/DeliveryMode.h>
```

Data Structures

- class **cms::Message**

  Root of all messages.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.60 src/main/activemq/commands/MessageAck.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageAck**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.61 src/main/activemq/commands/MessageDispatch.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageDispatch**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.62 src/main/activemq/commands/MessageDispatchNotification.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageDispatchNotification**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.63 src/main/activemq/commands/MessageId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessageId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.64 src/main/activemq/commands/MessagePull.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::MessagePull**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.65 src/main/activemq/commands/NetworkBridgeFilter.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::NetworkBridgeFilter**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.66 src/main/activemq/commands/PartialCommand.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::PartialCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.67 src/main/activemq/commands/ProducerAck.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerAck**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.68 src/main/activemq/commands/ProducerId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.69 src/main/activemq/commands/ProducerInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/BrokerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ProducerInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.70 src/main/activemq/commands/RemoveInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.71 src/main/activemq/commands/RemoveSubscriptionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::RemoveSubscriptionInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.72 src/main/activemq/commands/ReplayCommand.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ReplayCommand**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.73 src/main/activemq/commands/Response.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::Response**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.74 src/main/activemq/commands/SessionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.75 src/main/activemq/commands/SessionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/SessionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SessionInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.76 src/main/activemq/commands/ShutdownInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::ShutdownInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.77 src/main/activemq/commands/SubscriptionInfo.h File Reference

```
#include <activemq/commands/ActiveMQDestination.h>
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::SubscriptionInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.78 src/main/activemq/commands/TransactionId.h File Reference

```
#include <activemq/commands/BaseDataStructure.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.79 src/main/activemq/commands/TransactionInfo.h File Reference

```
#include <activemq/commands/BaseCommand.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::TransactionInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.80 src/main/activemq/commands/WireFormatInfo.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/BaseCommand.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <vector>
```

Data Structures

- class **activemq::commands::WireFormatInfo**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.81 src/main/activemq/commands/XATransactionId.h File Reference

```
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <vector>
```

Data Structures

- class **activemq::commands::XATransactionId**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**

7.82 src/main/activemq/core/ActiveMQAckHandler.h File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
```

Data Structures

- class **activemq::core::ActiveMQAckHandler**

  Interface class that is used to give CMS Messages an interface to Ack themselves with.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.83 src/main/activemq/core/ActiveMQConnection.h File Reference

```
#include <cms/Connection.h>
#include <activemq/util/Config.h>
#include <activemq/core/ActiveMQConnectionSupport.h>
#include <activemq/core/ActiveMQConnectionMetaData.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnection**

  Concrete connection used for all connectors to the ActiveMQ broker.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.84 src/main/activemq/core/ActiveMQConnectionFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/ConnectionFactory.h>
#include <cms/Connection.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionFactory**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.85 src/main/activemq/core/ActiveMQConnectionMetaData.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **activemq::core::ActiveMQConnectionMetaData**

  *This class houses all the various settings and information that is used by an instance of an **ActiveMQConnection** (p. 190) class.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.86 src/main/activemq/core/ActiveMQConnectionSupport.h File Reference

```
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConnectionSupport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.87 src/main/activemq/core/ActiveMQConstants.h File Reference

```
#include <string>
#include <map>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::core::ActiveMQConstants**

  Class holding constant values for various ActiveMQ specific things Each constant is defined as an enumeration and has functions that convert back an forth between string and enum values.

- class **activemq::core::ActiveMQConstants::StaticInitializer**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.88 src/main/activemq/core/ActiveMQConsumer.h File Reference

```
#include <cms/MessageConsumer.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/core/ActiveMQAckHandler.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQConsumer**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.89 src/main/activemq/core/ActiveMQProducer.h File Reference

```
#include <cms/MessageProducer.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/DeliveryMode.h>
#include <activemq/util/Config.h>
#include <activemq/util/MemoryUsage.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQProducer**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.90 src/main/activemq/core/ActiveMQSession.h File Reference

```
#include <cms/Session.h>
#include <cms/ExceptionListener.h>
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/core/ActiveMQTransactionContext.h>
#include <activemq/commands/ActiveMQTempDestination.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/core/Dispatcher.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/Properties.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::core::ActiveMQSession**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.91 src/main/activemq/core/ActiveMQSessionExecutor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/core/MessageDispatchChannel.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/threads/Task.h>
#include <activemq/threads/TaskRunner.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::ActiveMQSessionExecutor**

  Delegate dispatcher for a single session.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.92 src/main/activemq/core/ActiveMQTransactionContext.h File Reference

```
#include <memory>
#include <cms/Message.h>
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/LocalTransactionId.h>
#include <activemq/core/Synchronization.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::core::ActiveMQTransactionContext**

  Transaction Management class, hold messages that are to be redelivered upon a request to rollback.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.93 src/main/activemq/core/DispatchData.h File Reference

```
#include <stdlib.h>
#include <memory>
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/Message.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::DispatchData**

  Simple POCO that contains the information necessary to route a message to a specified consumer.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.94 src/main/activemq/core/Dispatcher.h File Reference

```
#include <activemq/commands/MessageDispatch.h>
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::Dispatcher**

  Interface for an object responsible for dispatching messages to consumers.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.95 src/main/activemq/core/MessageDispatchChannel.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/MessageDispatch.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::core::MessageDispatchChannel**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.96 src/main/activemq/core/Synchronization.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::core::Synchronization**

  Transacted Object ***Synchronization*** *(p. 2516), used to sync the events of a Transaction with the items in the Transaction.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::core**

7.97 src/main/activemq/exceptions/ActiveMQException.h File Reference

```
#include <activemq/util/Config.h>
#include <cms/CMSException.h>
#include <decaf/lang/Exception.h>
#include <activemq/exceptions/ExceptionDefines.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class **activemq::exceptions::ActiveMQException**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::exceptions**

7.98 src/main/activemq/exceptions/BrokerException.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/commands/BrokerError.h>
#include <sstream>
```

Data Structures

- class **activemq::exceptions::BrokerException**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::exceptions**

7.99 src/main/activemq/exceptions/ExceptionDefines.h File Reference

Defines

- #define **AMQ_CATCH_RETHROW**(type)

  Macro for catching and re-throwing an exception of a given type.

- #define **AMQ_CATCH_EXCEPTION_CONVERT**(sourceType, targetType)

  Macro for catching an exception of one type and then re-throwing as another type.

- #define **AMQ_CATCHALL_THROW**(type)

  A catch-all that throws a known exception.

- #define **AMQ_CATCHALL_NOTHROW**()

  A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

- #define **AMQ_CATCH_NOTHROW**(type)

  Macro for catching and re-throwing an exception of a given type.

- #define **AMQ_CATCH_ALL_THROW_CMSEXCEPTION**()

  Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

7.99.1 Define Documentation

7.99.1.1 #define AMQ_CATCH_ALL_THROW_CMSEXCEPTION()

Value:

```
catch( cms::CMSException& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw ex; \
    } catch( activemq::exceptions::ActiveMQException& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw ex.convertToCMSException(); \
    } catch( decaf::lang::Exception& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        activemq::exceptions::ActiveMQException amqEx( ex ); \
        throw amqEx.convertToCMSException(); \
    } catch( std::exception& ex ){ \
        throw cms::CMSException( ex.what(), NULL ); \
    } catch(...) { \
        throw cms::CMSException( "Caught Unknown Exception", NULL ); \
    }
```

Macro for catching an exception of one type and then re-throwing as a Basic CMSException, good for cases where the method isn't specific about what CMS Exceptions are thrown, bad if you need to throw an exception of MessageNotReadableException for instance.

Referenced by activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::acknowledge(), activemq::commands::ActiveMQTempTopic::destroy(), activemq::commands::ActiveMQTempQueue::destroy(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getBooleanProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getByteProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getDoubleProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getShortProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::getStringProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setBooleanProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setByteProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setDoubleProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setFloatProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setIntProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setLongProperty(), activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setShortProperty(), and activemq::commands::ActiveMQMessageTemplate< cms::ObjectMessage >::setStringProperty().

7.99.1.2 #define AMQ_CATCH_EXCEPTION_CONVERT(sourceType, targetType)

Value:

```
catch( sourceType& ex ){ \
        targetType target( ex ); \
        target.setMark( __FILE__, __LINE__ ); \
        throw target; \
    }
```

Macro for catching an exception of one type and then re-throwing as another type.

Parameters:

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray(), activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1(), and activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2().

7.99.1.3 #define AMQ_CATCH_NOTHROW(type)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
    }
```

Macro for catching and re-throwing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. ActiveMQException).

7.99.1.4 #define AMQ_CATCH_RETHROW(type)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw ex; \
    }
```

Macro for catching and re-throwing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. ActiveMQException).

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray(), activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1(), and activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2().

7.99.1.5 #define AMQ_CATCHALL_NOTHROW()

Value:

```
catch( ... ){ \
        activemq::exceptions::ActiveMQException ex( __FILE__, __LINE__, \
            "caught unknown exception, not rethrowing" ); \
    }
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

7.99.1.6 #define AMQ_CATCHALL_THROW(type)

Value:

```
catch( ... ){ \
        type ex( __FILE__, __LINE__, \
            "caught unknown exception" ); \
        throw ex; \
    }
```

A catch-all that throws a known exception.

Parameters:

type the type of exception to be thrown.

Referenced by activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::looseMarshalObjectArray(), activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray1(), and activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller::tightMarshalObjectArray2().

7.100 src/main/decaf/lang/exceptions/ExceptionDefines.h File Reference

Defines

- #define **DECAF_CATCH_RETHROW**(type)

  Macro for catching and rethrowing an exception of a given type.

- #define **DECAF_CATCH_EXCEPTION_CONVERT**(sourceType, targetType)

  Macro for catching an exception of one type and then rethrowing as another type.

- #define **DECAF_CATCHALL_THROW**(type)

  A catch-all that throws a known exception.

- #define **DECAF_CATCHALL_NOTHROW**()

  A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

- #define **DECAF_CATCH_NOTHROW**(type)

  Macro for catching and rethrowing an exception of a given type.

7.100.1 Define Documentation

7.100.1.1 #define DECAF_CATCH_EXCEPTION_CONVERT(sourceType, targetType)

Value:

```
catch( sourceType& ex ){ \
        targetType target( &ex ); \
        target.setMark( __FILE__, __LINE__ ); \
        throw target; \
    }
```

Macro for catching an exception of one type and then rethrowing as another type.

Parameters:

sourceType the type of the exception to be caught.

targetType the type of the exception to be thrown.

7.100.1.2 #define DECAF_CATCH_NOTHROW(type)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
    }
```

Macro for catching and rethrowing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. Exception).

Referenced by decaf::io::FilterInputStream::~FilterInputStream(), decaf::io::FilterOutputStream::~FilterOutputStream(), and decaf::util::logging::StreamHandler::~StreamHandler().

7.100.1.3 #define DECAF_CATCH_RETHROW(type)

Value:

```
catch( type& ex ){ \
        ex.setMark( __FILE__, __LINE__ ); \
        throw ex; \
    }
```

Macro for catching and rethrowing an exception of a given type.

Parameters:

type The type of the exception to throw (e.g. Exception).

Referenced by decaf::io::FilterInputStream::available(), decaf::io::FilterOutputStream::close(), decaf::io::FilterInputStream::close(), decaf::io::FilterOutputStream::flush(), decaf::util::concurrent::Lock::lock(), decaf::util::concurrent::Lock::Lock(), decaf::util::logging::StreamHandler::publish(), decaf::io::FilterInputStream::read(), decaf::io::FilterInputStream::reset(), decaf::io::FilterInputStream::skip(), decaf::util::concurrent::Lock::unlock(), decaf::io::FilterOutputStream::write(), and decaf::util::concurrent::Lock::~Lock().

7.100.1.4 #define DECAF_CATCHALL_NOTHROW()

Value:

```
catch( ... ){ \
        lang::Exception ex( __FILE__, __LINE__, \
            "caught unknown exception, not rethrowing" ); \
    }
```

A catch-all that does not throw an exception, one use would be to catch any exception in a destructor and mark it, but not throw so that cleanup would continue as normal.

Referenced by decaf::io::FilterInputStream::mark(), decaf::io::FilterInputStream::markSupported(), decaf::io::FilterInputStream::~FilterInputStream(), decaf::io::FilterOutputStream::~FilterOutputStream(), and decaf::util::logging::StreamHandler::~StreamHandler().

7.100.1.5 #define DECAF_CATCHALL_THROW(type)

Value:

```
catch( ... ){ \
        type ex( __FILE__, __LINE__, \
            "caught unknown exception" ); \
        throw ex; \
    }
```

A catch-all that throws a known exception.

Parameters:

type the type of exception to be thrown.

Referenced by decaf::io::FilterInputStream::available(), decaf::io::FilterOutputStream::close(), decaf::io::FilterInputStream::close(), decaf::io::FilterOutputStream::flush(), decaf::util::concurrent::Lock::lock(), decaf::util::concurrent::Lock::Lock(), decaf::util::logging::StreamHandler::publish(), decaf::io::FilterInputStream::read(), decaf::io::FilterInputStream::reset(), decaf::io::FilterInputStream::skip(), decaf::util::concurrent::Lock::unlock(), decaf::io::FilterOutputStream::write(), and decaf::util::concurrent::Lock::~Lock().

7.101 src/main/activemq/io/LoggingInputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterInputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **activemq::io::LoggingInputStream**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::io**

7.102 src/main/activemq/io/LoggingOutputStream.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/io/FilterOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
```

Data Structures

- class **activemq::io::LoggingOutputStream**

  OutputStream filter that just logs the data being written.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::io**

7.103 src/main/activemq/library/ActiveMQCPP.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::library::ActiveMQCPP**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::library**

7.104 src/main/activemq/state/CommandVisitor.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::CommandVisitor**

  Interface for an Object that can visit the various Command Objects that are sent from and to this client.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::commands**
- namespace **activemq::state**

7.105 src/main/activemq/state/CommandVisitorAdapter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/state/CommandVisitor.h>
#include <activemq/core/ActiveMQConstants.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ProducerInfo.h>
#include <activemq/commands/ConsumerInfo.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/commands/SessionId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/RemoveSubscriptionInfo.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageAck.h>
#include <activemq/commands/MessagePull.h>
#include <activemq/commands/TransactionInfo.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/ProducerAck.h>
#include <activemq/commands/MessageDispatch.h>
#include <activemq/commands/MessageDispatchNotification.h>
#include <activemq/commands/ControlCommand.h>
#include <activemq/commands/ConnectionError.h>
#include <activemq/commands/ConnectionControl.h>
#include <activemq/commands/ConsumerControl.h>
#include <activemq/commands/ShutdownInfo.h>
#include <activemq/commands/KeepAliveInfo.h>
#include <activemq/commands/FlushCommand.h>
#include <activemq/commands/BrokerError.h>
#include <activemq/commands/BrokerInfo.h>
#include <activemq/commands/RemoveInfo.h>
#include <activemq/commands/ReplayCommand.h>
#include <activemq/commands/Response.h>
```

Data Structures

- class **activemq::state::CommandVisitorAdapter**

  Default Implementation of a ***CommandVisitor*** *(*p. 885*) that returns NULL for all calls.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.106 src/main/activemq/state/ConnectionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionInfo.h>
#include <activemq/commands/DestinationInfo.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ConnectionState**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.107 src/main/activemq/state/ConnectionStateTracker.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConnectionId.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/state/CommandVisitorAdapter.h>
#include <activemq/state/ConnectionState.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <activemq/state/SessionState.h>
#include <activemq/state/TransactionState.h>
#include <activemq/state/Tracked.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::ConnectionStateTracker**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.108 src/main/activemq/state/ConsumerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ConsumerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ConsumerState**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.109 src/main/activemq/state/ProducerState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/ProducerInfo.h>
#include <decaf/lang/Pointer.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::ProducerState**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.110 src/main/activemq/state/SessionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/SessionInfo.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/state/ConsumerState.h>
#include <activemq/state/ProducerState.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/util/concurrent/ConcurrentStlMap.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::SessionState**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.111 src/main/activemq/state/Tracked.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::state::Tracked**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.112 src/main/activemq/state/TransactionState.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/TransactionId.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlList.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <string>
#include <memory>
```

Data Structures

- class **activemq::state::TransactionState**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::state**

7.113 src/main/activemq/threads/CompositeTask.h File Reference

`#include <activemq/util/Config.h>`

`#include <activemq/threads/Task.h>`

Data Structures

- class **activemq::threads::CompositeTask**

  Represents a single task that can be part of a set of Tasks that are contained in a ***CompositeTaskRunner*** *(p. 908).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.114 src/main/activemq/threads/CompositeTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTask.h>
#include <decaf/util/StlSet.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::CompositeTaskRunner**

  *A **Task** (p. 2520) Runner that can contain one or more CompositeTasks that are each checked for pending work and run if any is present in the order that the tasks were added.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.115 src/main/activemq/threads/DedicatedTaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/Task.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::threads::DedicatedTaskRunner**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.116 src/main/activemq/threads/Task.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::threads::Task**

  Represents a unit of work that requires one or more iterations to complete.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.117 src/main/activemq/threads/TaskRunner.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/Task.h>
```

Data Structures

- class **activemq::threads::TaskRunner**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::threads**

7.118 src/main/activemq/transport/AbstractTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportFactory.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::AbstractTransportFactory**

  *Abstract implementation of the **TransportFactory** (p. 2614) interface, providing the base functionality that's common to most of the **TransportFactory** (p. 2614) instances.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.119 src/main/activemq/transport/CompositeTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::transport::CompositeTransport**

  *A Composite **Transport** (p. 2608) is a **Transport** (p. 2608) implementation that is composed of several Transports.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.120 src/main/activemq/transport/correlator/FutureResponse.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::correlator::FutureResponse**

  A container that holds a response object.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.121 src/main/activemq/transport/correlator/ResponseCorrelator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/transport/correlator/FutureResponse.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <map>
#include <stdio.h>
```

Data Structures

- class **activemq::transport::correlator::ResponseCorrelator**

  *This type of **transport** (p. 67) filter is responsible for correlating asynchronous responses with requests.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::correlator**

7.122 src/main/activemq/transport/DefaultTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::DefaultTransportListener**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.123 src/main/activemq/transport/failover/BackupTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <memory>
```

Data Structures

- class **activemq::transport::failover::BackupTransport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.124 src/main/activemq/transport/failover/BackupTransportPool.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/BackupTransport.h>
#include <activemq/transport/failover/URIPool.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/IOException.h>
#include <decaf/util/StlList.h>
```

Data Structures

- class **activemq::transport::failover::BackupTransportPool**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.125 src/main/activemq/transport/failover/CloseTransportsTask.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/threads/CompositeTask.h>
#include <activemq/transport/Transport.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::CloseTransportsTask**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.126 src/main/activemq/transport/failover/FailoverTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/threads/TaskRunner.h>
#include <activemq/threads/CompositeTaskRunner.h>
#include <activemq/state/ConnectionStateTracker.h>
#include <activemq/transport/CompositeTransport.h>
#include <activemq/transport/failover/BackupTransportPool.h>
#include <activemq/transport/failover/CloseTransportsTask.h>
#include <activemq/transport/failover/FailoverTransportListener.h>
#include <activemq/transport/failover/URIPool.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/StlList.h>
#include <decaf/util/StlMap.h>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/atomic/AtomicReference.h>
#include <decaf/net/URI.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.127 src/main/activemq/transport/failover/FailoverTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/transport/Transport.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportFactory**

  *Creates an instance of a **FailoverTransport** (p. 1296).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.128 src/main/activemq/transport/failover/FailoverTransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::failover::FailoverTransportListener**

  *Utility class used by the **Transport** (p. 2608) to perform the work of responding to events from the active **Transport** (p. 2608).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.129 src/main/activemq/transport/failover/URIPool.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/net/URI.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
```

Data Structures

- class **activemq::transport::failover::URIPool**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::failover**

7.130 src/main/activemq/transport/IOTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Thread.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <memory>
```

Data Structures

- class **activemq::transport::IOTransport**

  *Implementation of the **Transport** (p. 2608) interface that performs marshaling of **commands** (p. 59) to IO streams.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.131 src/main/activemq/transport/logging/LoggingTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::logging::LoggingTransport**

  *A **transport** (p. 67) filter that logs **commands** (p. 59) as they are sent/received.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::logging**

7.132 src/main/activemq/transport/mock/InternalCommandListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
```

Data Structures

- class **activemq::transport::mock::InternalCommandListener**

  *Listens for Commands sent from the **MockTransport** (p. 1910).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.133 src/main/activemq/transport/mock/MockTransport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/transport/TransportListener.h>
#include <activemq/transport/DefaultTransportListener.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <activemq/transport/mock/InternalCommandListener.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/lang/Thread.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <cms/Message.h>
#include <map>
#include <set>
```

Data Structures

- class **activemq::transport::mock::MockTransport**

  *The **MockTransport** (p. 1910) defines a base level **Transport** (p. 2608) class that is intended to be used in place of an a regular protocol **Transport** (p. 2608) suck as TCP.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.134 src/main/activemq/transport/mock/MockTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <activemq/transport/AbstractTransportFactory.h>
```

Data Structures

- class **activemq::transport::mock::MockTransportFactory**

  Manufactures MockTransports, which are objects that read from input streams and write to output streams.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.135 src/main/activemq/transport/mock/ResponseBuilder.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/StlQueue.h>
```

Data Structures

- class **activemq::transport::mock::ResponseBuilder**

  Interface for all Protocols to implement that defines the behavior of the Broker in response to messages of that protocol.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::mock**

7.136 src/main/activemq/transport/tcp/TcpTransport.h File Reference

```
#include <activemq/io/LoggingInputStream.h>
#include <activemq/io/LoggingOutputStream.h>
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/net/Socket.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <memory>
```

Data Structures

- class **activemq::transport::tcp::TcpTransport**

  Implements a TCP/IP based ***transport*** *(p. 67) filter, this* ***transport*** *(p. 67) is meant to wrap an instance of an* ***IOTransport*** *(p. 1480).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.137 src/main/activemq/transport/tcp/TcpTransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/AbstractTransportFactory.h>
#include <activemq/exceptions/ActiveMQException.h>
```

Data Structures

- class **activemq::transport::tcp::TcpTransportFactory**

  Factory Responsible for creating the ***TcpTransport*** *(*p. 2532*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**
- namespace **activemq::transport::tcp**

7.138 src/main/activemq/transport/Transport.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/net/URI.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/commands/Response.h>
#include <cms/Startable.h>
#include <cms/Closeable.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::Transport**

  *Interface for a **transport** (p. 67) layer for command objects.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::transport**

7.139 src/main/activemq/transport/TransportFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <decaf/net/URI.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportFactory**

  Defines the interface for Factories that create Transports or TransportFilters.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.140 src/main/activemq/transport/TransportFilter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <activemq/transport/Transport.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/TransportListener.h>
#include <decaf/lang/Pointer.h>
#include <typeinfo>
```

Data Structures

- class **activemq::transport::TransportFilter**

  *A filter on the **transport** (p. 67) layer.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.141 src/main/activemq/transport/TransportListener.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::transport::TransportListener**

  *A listener of asynchronous **exceptions** (p. 62) from a command **transport** (p. 67) object.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.142 src/main/activemq/transport/TransportRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/transport/TransportFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::transport::TransportRegistry**

  *Registry of all **Transport** (p. 2608) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::transport**

7.143 src/main/activemq/util/ActiveMQProperties.h File Reference

```
#include <map>
#include <string>
#include <sstream>
#include <activemq/util/Config.h>
#include <cms/CMSProperties.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::util::ActiveMQProperties**

  *Implementation of the CMSProperties interface that delegates to a **decaf::util::Properties** (p. 2140) object.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.144 src/main/activemq/util/CompositeData.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/net/URI.h>
#include <decaf/net/URISyntaxException.h>
```

Data Structures

- class **activemq::util::CompositeData**

  Represents a Composite URI.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.145 src/main/activemq/util/Config.h File Reference

Defines

- #define **AMQCPP_API**

7.145.1 Define Documentation

7.145.1.1 #define AMQCPP_API

7.146 src/main/cms/Config.h File Reference

Defines

- #define **CMS_API**

7.146.1 Define Documentation

7.146.1.1 #define CMS_API

7.147 src/main/decaf/util/Config.h File Reference

Defines

- #define **DECAF_API**
- #define **DECAF_UNUSED**

7.147.1 Define Documentation

7.147.1.1 #define DECAF_API

7.147.1.2 #define DECAF_UNUSED

7.148 src/main/activemq/util/LongSequenceGenerator.h File Reference

```
#include <activemq/util/Config.h>
```

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::LongSequenceGenerator**

  This class is used to generate a sequence of long long values that are incremented each time a new value is requested.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.149 src/main/activemq/util/MemoryUsage.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/Usage.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **activemq::util::MemoryUsage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.150 src/main/activemq/util/PrimitiveList.h File Reference

```
#include <string>
#include <vector>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <stdio.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveList**

  List of primitives.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.151 src/main/activemq/util/PrimitiveMap.h File Reference

```
#include <string>
#include <vector>
#include <activemq/util/Config.h>
#include <decaf/util/Config.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveValueConverter.h>
```

Data Structures

- class **activemq::util::PrimitiveMap**

  Map of named primitives.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.152 src/main/activemq/util/PrimitiveValueConverter.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <string>
```

Data Structures

- class **activemq::util::PrimitiveValueConverter**

  *Class controls the conversion of data contained in a **PrimitiveValueNode** (p. 2070) from one type to another.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.153 src/main/activemq/util/PrimitiveValueNode.h File Reference

```
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Map.h>
#include <decaf/util/List.h>
```

Data Structures

- class **activemq::util::PrimitiveValueNode**

  Class that wraps around a single value of one of the many types.

- union **activemq::util::PrimitiveValueNode::PrimitiveValue**

  Define a union type comprised of the various types.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.154 src/main/activemq/util/URISupport.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/CompositeData.h>
#include <decaf/util/Properties.h>
#include <decaf/util/StlList.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::util::URISupport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.155 src/main/activemq/util/Usage.h File Reference

```
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::util::Usage**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::util**

7.156 src/main/activemq/wireformat/MarshalAware.h File Reference

```
#include <vector>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::MarshalAware**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.157 src/main/activemq/wireformat/openwire/marshal/BaseDataStrea File Reference

```
#include <activemq/wireformat/openwire/marshal/DataStreamMarshaller.h>
#include <activemq/wireformat/openwire/utils/HexTable.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::BaseDataStreamMarshaller**

  Base class for all Marshallers that ***marshal*** *(p. 76) DataStructures to and from the wire using the OpenWire protocol.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.158 src/main/activemq/wireformat/openwire/marshal/DataStreamM File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::DataStreamMarshaller**

  *Base class for all classes that **marshal** (p. 76) **commands** (p. 59) for Openwire.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.159 src/main/activemq/wireformat/openwire/marshal/PrimitiveType File Reference

```
#include <cms/CMSException.h>
#include <activemq/util/Config.h>
#include <activemq/util/PrimitiveValueNode.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/util/PrimitiveList.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/IOException.h>
#include <string>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::PrimitiveTypesMarshaller**

  This class wraps the functionality needed to ***marshal*** *(p. 76) a primitive map to the Openwire Format's expectation of what the map looks like on the wire.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.160 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBlobMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 155).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.161 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBlobMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 159).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.162 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQBlobMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBlobMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBlobMessageMarshaller** (p. 151).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.163 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQBytesMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 182).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.164 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQBytesMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQBytesMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 186).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.165 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQBytesMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQBytesMessageMarshaller** (p. 178).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.166 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 244).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.167 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 248).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.168 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQDestinationMarshaller** (p. 240).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.169 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMapMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 271).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.170 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQMapMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMapMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 275).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.171 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMapMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMapMessageMarshaller** (p. 267).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.172 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 286).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.173 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 290).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.174 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQMessageMarshaller** (p. 282).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.175 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQObjectMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 317).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.176 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQObjectMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 321).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.177 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQObjectMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQObjectMessageMarshaller** (p. 313).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.178 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQQueueMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 345).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.179 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQQueueMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQQueueMarshaller*** *(p. 349).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.180 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQQueueMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQQueueMarshaller** (p. 341).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.181 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQStreamMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 391).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.182 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQStreamMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 395).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.183 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQS File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQStreamMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQStreamMessageMarshaller** (p. 387).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.184 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempDestinationMarshaller**

  Marshaling code for Open Wire Format for ***ActiveMQTempDestinationMarshaller*** *(p. 407).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.185 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempDestinationMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempDestinationMarshaller** (p. 411).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.186 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempDestinationMarshaller**

  Marshaling code for Open Wire Format for ***ActiveMQTempDestinationMarshaller*** *(p. 403).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.187 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQ File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 424).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.188 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 428).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.189 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempQueueMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempQueueMarshaller** (p. 420).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.190 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQTempTopicMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTempTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 441).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.191 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTempTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 445).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.192 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQTempDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTempTopicMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTempTopicMarshaller** (p. 437).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.193 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTextMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 457).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.194 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQTextMessageMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTextMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 461).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.195 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/MessageMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTextMessageMarshaller**

  *Marshaling code for Open Wire Format for **ActiveMQTextMessageMarshaller** (p. 453).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.196 src/main/activemq/wireformat/openwire/marshal/v1/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ActiveMQTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTopicMarshaller*** *(p. 473).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.197 src/main/activemq/wireformat/openwire/marshal/v2/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ActiveMQTopicMarshaller**
  Marshaling code for Open Wire Format for ***ActiveMQTopicMarshaller*** *(p. 477).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.198 src/main/activemq/wireformat/openwire/marshal/v3/ActiveMQT File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ActiveMQDestinationMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ActiveMQTopicMarshaller**
  *Marshaling code for Open Wire Format for **ActiveMQTopicMarshaller** (p. 469).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.199 src/main/activemq/wireformat/openwire/marshal/v1/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BaseCommandMarshaller**
  Marshaling code for Open Wire Format for ***BaseCommandMarshaller*** *(p. 521).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.200 src/main/activemq/wireformat/openwire/marshal/v2/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BaseCommandMarshaller**

  Marshaling code for Open Wire Format for ***BaseCommandMarshaller*** *(*p. 528*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.201 src/main/activemq/wireformat/openwire/marshal/v3/BaseComm File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BaseCommandMarshaller**
  Marshaling code for Open Wire Format for ***BaseCommandMarshaller*** *(p. 514).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.202 src/main/activemq/wireformat/openwire/marshal/v1/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerIdMarshaller**

  *Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 599).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.203 src/main/activemq/wireformat/openwire/marshal/v2/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerIdMarshaller**
  *Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 603).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.204 src/main/activemq/wireformat/openwire/marshal/v3/BrokerIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerIdMarshaller**

  *Marshaling code for Open Wire Format for **BrokerIdMarshaller** (p. 595).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.205 src/main/activemq/wireformat/openwire/marshal/v1/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::BrokerInfoMarshaller**
  Marshaling code for Open Wire Format for ***BrokerInfoMarshaller*** *(*p. 619*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.206 src/main/activemq/wireformat/openwire/marshal/v2/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::BrokerInfoMarshaller**

  Marshaling code for Open Wire Format for ***BrokerInfoMarshaller*** *(p. 623).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.207 src/main/activemq/wireformat/openwire/marshal/v3/BrokerInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::BrokerInfoMarshaller**
  *Marshaling code for Open Wire Format for **BrokerInfoMarshaller** (p. 615).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.208 src/main/activemq/wireformat/openwire/marshal/v1/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionControlMarshaller**

  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(p. 954).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.209 src/main/activemq/wireformat/openwire/marshal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionControlMarshaller**

  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(*p. 958*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.210 src/main/activemq/wireformat/openwire/marshal/v3/ConnectionControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionControlMarshaller**

  Marshaling code for Open Wire Format for ***ConnectionControlMarshaller*** *(p. 950).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.211 src/main/activemq/wireformat/openwire/marshal/v1/Connectior File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 970).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.212 src/main/activemq/wireformat/openwire/marshal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 974).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.213 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionErrorMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionErrorMarshaller*** *(p. 966).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.214 src/main/activemq/wireformat/openwire/marshal/v1/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 989).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.215 src/main/activemq/wireformat/openwire/marshal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 993).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.216 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionIdMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionIdMarshaller*** *(p. 985).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.217 src/main/activemq/wireformat/openwire/marshal/v1/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(p. 1007).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.218 src/main/activemq/wireformat/openwire/marshal/v2/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(p. 1011).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.219 src/main/activemq/wireformat/openwire/marshal/v3/Connection File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConnectionInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConnectionInfoMarshaller*** *(*p. *1003**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.220 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerControlMarshaller**

  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1037).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.221 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerC File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerControlMarshaller**

  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1041).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.222 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerControlMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerControlMarshaller**

  Marshaling code for Open Wire Format for ***ConsumerControlMarshaller*** *(p. 1033).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.223 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(p. 1054).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.224 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(p. 1058).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.225 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerIdMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerIdMarshaller*** *(*p. *1050**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.226 src/main/activemq/wireformat/openwire/marshal/v1/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1074).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.227 src/main/activemq/wireformat/openwire/marshal/v2/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1078).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.228 src/main/activemq/wireformat/openwire/marshal/v3/ConsumerI File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ConsumerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ConsumerInfoMarshaller*** *(p. 1070).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.229 src/main/activemq/wireformat/openwire/marshal/v1/ControlCo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ControlCommandMarshaller**

  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(*p. *1087**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.230 src/main/activemq/wireformat/openwire/marshal/v2/ControlCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ControlCommandMarshaller**

  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(*p. *1095**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.231 src/main/activemq/wireformat/openwire/marshal/v3/ControlCo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ControlCommandMarshaller**

  Marshaling code for Open Wire Format for ***ControlCommandMarshaller*** *(*p. *1091**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.232 src/main/activemq/wireformat/openwire/marshal/v1/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataArrayResponseMarshaller**

  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(*p. *1108**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.233 src/main/activemq/wireformat/openwire/marshal/v2/DataArray File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataArrayResponseMarshaller**

  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(*p. *1112**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.234 src/main/activemq/wireformat/openwire/marshal/v3/DataArrayResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataArrayResponseMarshaller**

  Marshaling code for Open Wire Format for ***DataArrayResponseMarshaller*** *(*p. *1104**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.235 src/main/activemq/wireformat/openwire/marshal/v1/DataRespo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DataResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(*p. *1139**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.236 src/main/activemq/wireformat/openwire/marshal/v2/DataRespo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DataResponseMarshaller**

  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(*p. *1143**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.237 src/main/activemq/wireformat/openwire/marshal/v3/DataRespo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DataResponseMarshaller**
  Marshaling code for Open Wire Format for ***DataResponseMarshaller*** *(*p. *1135**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.238 src/main/activemq/wireformat/openwire/marshal/v1/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** (p. *1203*).

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.239 src/main/activemq/wireformat/openwire/marshal/v2/Destinatio File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** *(p. 1207).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.240 src/main/activemq/wireformat/openwire/marshal/v3/DestinationInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DestinationInfoMarshaller**
  Marshaling code for Open Wire Format for ***DestinationInfoMarshaller*** (p. *1199*).

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.241 src/main/activemq/wireformat/openwire/marshal/v1/DiscoveryE File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1221).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.242 src/main/activemq/wireformat/openwire/marshal/v2/DiscoveryE File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1225).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.243 src/main/activemq/wireformat/openwire/marshal/v3/DiscoveryE File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::DiscoveryEventMarshaller**
  Marshaling code for Open Wire Format for ***DiscoveryEventMarshaller*** *(p. 1217).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.244 src/main/activemq/wireformat/openwire/marshal/v1/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ExceptionResponseMarshaller**

  *Marshaling code for Open Wire Format for **ExceptionResponseMarshaller** (p. 1283).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.245 src/main/activemq/wireformat/openwire/marshal/v2/ExceptionR File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ExceptionResponseMarshaller**

  Marshaling code for Open Wire Format for ***ExceptionResponseMarshaller*** *(*p. *1287**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.246 src/main/activemq/wireformat/openwire/marshal/v3/ExceptionResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ExceptionResponseMarshaller**

  Marshaling code for Open Wire Format for ***ExceptionResponseMarshaller*** *(*p. *1279**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.247 src/main/activemq/wireformat/openwire/marshal/v1/FlushComm File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1360).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.248 src/main/activemq/wireformat/openwire/marshal/v2/FlushComm File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1368).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.249 src/main/activemq/wireformat/openwire/marshal/v3/FlushComm File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::FlushCommandMarshaller**
  Marshaling code for Open Wire Format for ***FlushCommandMarshaller*** *(p. 1364).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.250 src/main/activemq/wireformat/openwire/marshal/v1/IntegerResponseMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::IntegerResponseMarshaller**
  Marshaling code for Open Wire Format for ***IntegerResponseMarshaller*** *(p. 1449).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.251 src/main/activemq/wireformat/openwire/marshal/v2/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::IntegerResponseMarshaller**
  *Marshaling code for Open Wire Format for **IntegerResponseMarshaller** (p. 1445).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.252 src/main/activemq/wireformat/openwire/marshal/v3/IntegerRes File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/ResponseMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::IntegerResponseMarshaller**
  Marshaling code for Open Wire Format for ***IntegerResponseMarshaller*** *(*p. *1453**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.253 src/main/activemq/wireformat/openwire/marshal/v1/JournalQu File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(p. 1499).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.254 src/main/activemq/wireformat/openwire/marshal/v2/JournalQueueAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(p. 1495).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.255 src/main/activemq/wireformat/openwire/marshal/v3/JournalQu File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalQueueAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalQueueAckMarshaller*** *(*p. *1503)**.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.256 src/main/activemq/wireformat/openwire/marshal/v1/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTopicAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1520).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.257 src/main/activemq/wireformat/openwire/marshal/v2/JournalTop File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTopicAckMarshaller**
  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1512).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.258 src/main/activemq/wireformat/openwire/marshal/v3/JournalTopicAckMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTopicAckMarshaller**

  Marshaling code for Open Wire Format for ***JournalTopicAckMarshaller*** *(p. 1516).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.259 src/main/activemq/wireformat/openwire/marshal/v1/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTraceMarshaller**
  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1535).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.260 src/main/activemq/wireformat/openwire/marshal/v2/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTraceMarshaller**
  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1527).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.261 src/main/activemq/wireformat/openwire/marshal/v3/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTraceMarshaller**
  Marshaling code for Open Wire Format for ***JournalTraceMarshaller*** *(p. 1531).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.262 src/main/activemq/wireformat/openwire/marshal/v1/JournalTransactionMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::JournalTransactionMarshaller**

  Marshaling code for Open Wire Format for ***JournalTransactionMarshaller*** *(p. 1551).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.263 src/main/activemq/wireformat/openwire/marshal/v2/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::JournalTransactionMarshaller**

  *Marshaling code for Open Wire Format for **JournalTransactionMarshaller** (p. 1543).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.264 src/main/activemq/wireformat/openwire/marshal/v3/JournalTra File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::JournalTransactionMarshaller**

  Marshaling code for Open Wire Format for ***JournalTransactionMarshaller*** *(*p. 1547*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.265 src/main/activemq/wireformat/openwire/marshal/v1/KeepAlivel File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::KeepAliveInfoMarshaller**
  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(*p. *1566**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.266 src/main/activemq/wireformat/openwire/marshal/v2/KeepAliveI File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::KeepAliveInfoMarshaller**
  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(p. 1562).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.267 src/main/activemq/wireformat/openwire/marshal/v3/KeepAlivel File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::KeepAliveInfoMarshaller**
  Marshaling code for Open Wire Format for ***KeepAliveInfoMarshaller*** *(p. 1558).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.268 src/main/activemq/wireformat/openwire/marshal/v1/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1582).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.269 src/main/activemq/wireformat/openwire/marshal/v2/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1586).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.270 src/main/activemq/wireformat/openwire/marshal/v3/LastPartial File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LastPartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***LastPartialCommandMarshaller*** *(p. 1578).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.271 src/main/activemq/wireformat/openwire/marshal/v1/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::LocalTransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***LocalTransactionIdMarshaller*** *(p. 1612).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.272 src/main/activemq/wireformat/openwire/marshal/v2/LocalTransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::LocalTransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***LocalTransactionIdMarshaller*** *(p. 1604).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.273 src/main/activemq/wireformat/openwire/marshal/v3/LocalTrans File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::LocalTransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **LocalTransactionIdMarshaller** (p. 1608).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.274 src/main/activemq/wireformat/openwire/marshal/v1/MarshallerFactory.h File Reference

`#include <activemq/wireformat/openwire/OpenWireFormat.h>`

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.275 src/main/activemq/wireformat/openwire/marshal/v2/Marshaller File Reference

`#include <activemq/wireformat/openwire/OpenWireFormat.h>`

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.276 src/main/activemq/wireformat/openwire/marshal/v3/MarshallerFactory.h File Reference

`#include <activemq/wireformat/openwire/OpenWireFormat.h>`

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MarshallerFactory**

  Used to create marshallers for a specific version of the wire protocol.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.277 src/main/activemq/wireformat/openwire/marshal/v1/MessageAc File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageAckMarshaller**
  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(*p. *1791**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.278 src/main/activemq/wireformat/openwire/marshal/v2/MessageAc File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageAckMarshaller**
  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(*p. *1783**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.279 src/main/activemq/wireformat/openwire/marshal/v3/MessageAc File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageAckMarshaller**
  Marshaling code for Open Wire Format for ***MessageAckMarshaller*** *(*p. *1787**).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.280 src/main/activemq/wireformat/openwire/marshal/v1/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(*p. *1819**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.281 src/main/activemq/wireformat/openwire/marshal/v2/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(*p. *1811**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.282 src/main/activemq/wireformat/openwire/marshal/v3/MessageDispatchMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchMarshaller**

  Marshaling code for Open Wire Format for ***MessageDispatchMarshaller*** *(*p. *1815**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.283 src/main/activemq/wireformat/openwire/marshal/v1/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageDispatchNotificationMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1833).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.284 src/main/activemq/wireformat/openwire/marshal/v2/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageDispatchNotificationMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1829).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.285 src/main/activemq/wireformat/openwire/marshal/v3/MessageDi File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageDispatchNotificationMarshaller**

  *Marshaling code for Open Wire Format for **MessageDispatchNotificationMarshaller** (p. 1837).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.286 src/main/activemq/wireformat/openwire/marshal/v1/MessageId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(p. 1848).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.287 src/main/activemq/wireformat/openwire/marshal/v2/MessageId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(*p. *1856**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.288 src/main/activemq/wireformat/openwire/marshal/v3/MessageId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageIdMarshaller**

  Marshaling code for Open Wire Format for ***MessageIdMarshaller*** *(*p. *1852**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.289 src/main/activemq/wireformat/openwire/marshal/v1/MessageMa File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(*p. *1871**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.290 src/main/activemq/wireformat/openwire/marshal/v2/MessageMa File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(p. 1861).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.291 src/main/activemq/wireformat/openwire/marshal/v3/MessageM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessageMarshaller**

  Marshaling code for Open Wire Format for ***MessageMarshaller*** *(*p. *1866**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.292 src/main/activemq/wireformat/openwire/marshal/v1/MessagePu File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::MessagePullMarshaller**
  *Marshaling code for Open Wire Format for **MessagePullMarshaller*** (p. *1906*).

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.293 src/main/activemq/wireformat/openwire/marshal/v2/MessagePu File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::MessagePullMarshaller**
  *Marshaling code for Open Wire Format for **MessagePullMarshaller** (p. 1898).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.294 src/main/activemq/wireformat/openwire/marshal/v3/MessagePu File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::MessagePullMarshaller**
  *Marshaling code for Open Wire Format for **MessagePullMarshaller*** (p. *1902*).

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.295 src/main/activemq/wireformat/openwire/marshal/v1/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::NetworkBridgeFilterMarshaller**

  *Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller*** (p. *1935*).

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.296 src/main/activemq/wireformat/openwire/marshal/v2/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::NetworkBridgeFilterMarshaller**

  *Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1931).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.297 src/main/activemq/wireformat/openwire/marshal/v3/NetworkBr File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::NetworkBridgeFilterMarshaller**

  *Marshaling code for Open Wire Format for **NetworkBridgeFilterMarshaller** (p. 1927).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.298 src/main/activemq/wireformat/openwire/marshal/v1/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(p. 2007).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.299 src/main/activemq/wireformat/openwire/marshal/v2/PartialCon File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(*p. *1999**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.300 src/main/activemq/wireformat/openwire/marshal/v3/PartialCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::PartialCommandMarshaller**

  Marshaling code for Open Wire Format for ***PartialCommandMarshaller*** *(p. 2003).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.301 src/main/activemq/wireformat/openwire/marshal/v1/ProducerA File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerAckMarshaller**
  *Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2094).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.302 src/main/activemq/wireformat/openwire/marshal/v2/ProducerA File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerAckMarshaller**

  *Marshaling code for Open Wire Format for **ProducerAckMarshaller** (p. 2098).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.303 src/main/activemq/wireformat/openwire/marshal/v3/ProducerA File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerAckMarshaller**
  Marshaling code for Open Wire Format for ***ProducerAckMarshaller*** *(*p. 2090*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.304 src/main/activemq/wireformat/openwire/marshal/v1/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerIdMarshaller**

  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(*p. *2118**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.305 src/main/activemq/wireformat/openwire/marshal/v2/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerIdMarshaller**
  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(p. 2110).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.306 src/main/activemq/wireformat/openwire/marshal/v3/ProducerId File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerIdMarshaller**

  Marshaling code for Open Wire Format for ***ProducerIdMarshaller*** *(*p. *2114**).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.307 src/main/activemq/wireformat/openwire/marshal/v1/ProducerIn File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ProducerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(*p. 2135*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.308 src/main/activemq/wireformat/openwire/marshal/v2/ProducerIn File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ProducerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(p. 2127).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.309 src/main/activemq/wireformat/openwire/marshal/v3/ProducerIn File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ProducerInfoMarshaller**
  Marshaling code for Open Wire Format for ***ProducerInfoMarshaller*** *(*p. 2131*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.310 src/main/activemq/wireformat/openwire/marshal/v1/RemoveInf File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveInfoMarshaller**
  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2185).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.311 src/main/activemq/wireformat/openwire/marshal/v2/RemoveInf File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveInfoMarshaller**
  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2181).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.312 src/main/activemq/wireformat/openwire/marshal/v3/RemoveInf File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveInfoMarshaller**
  Marshaling code for Open Wire Format for ***RemoveInfoMarshaller*** *(p. 2189).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.313 src/main/activemq/wireformat/openwire/marshal/v1/RemoveSu File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::RemoveSubscriptionInfoMarshaller**

  *Marshaling code for Open Wire Format for **RemoveSubscriptionInfoMarshaller** (p. 2206).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.314 src/main/activemq/wireformat/openwire/marshal/v2/RemoveSubscriptionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::RemoveSubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveSubscriptionInfoMarshaller*** *(*p. 2198*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.315 src/main/activemq/wireformat/openwire/marshal/v3/RemoveSu File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::RemoveSubscriptionInfoMarshaller**

  Marshaling code for Open Wire Format for ***RemoveSubscriptionInfoMarshaller*** *(*p. 2202*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.316 src/main/activemq/wireformat/openwire/marshal/v1/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ReplayCommandMarshaller**

  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2222).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.317 src/main/activemq/wireformat/openwire/marshal/v2/ReplayCon File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ReplayCommandMarshaller**

  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2214).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.318 src/main/activemq/wireformat/openwire/marshal/v3/ReplayCommandMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ReplayCommandMarshaller**

  Marshaling code for Open Wire Format for ***ReplayCommandMarshaller*** *(p. 2218).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.319 src/main/activemq/wireformat/openwire/marshal/v1/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ResponseMarshaller**

  Marshaling code for Open Wire Format for ***ResponseMarshaller*** *(p. 2251).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.320 src/main/activemq/wireformat/openwire/marshal/v2/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ResponseMarshaller**

  Marshaling code for Open Wire Format for ***ResponseMarshaller*** (p. *2241*)*.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.321 src/main/activemq/wireformat/openwire/marshal/v3/ResponseM File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ResponseMarshaller**
  Marshaling code for Open Wire Format for ***ResponseMarshaller*** *(p. 2246).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.322 src/main/activemq/wireformat/openwire/marshal/v1/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionIdMarshaller**

  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(*p. 2288*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.323 src/main/activemq/wireformat/openwire/marshal/v2/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionIdMarshaller**

  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(*p. 2296*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.324 src/main/activemq/wireformat/openwire/marshal/v3/SessionIdM File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionIdMarshaller**

  Marshaling code for Open Wire Format for ***SessionIdMarshaller*** *(*p. 2292*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.325 src/main/activemq/wireformat/openwire/marshal/v1/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SessionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(p. 2304).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.326 src/main/activemq/wireformat/openwire/marshal/v2/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SessionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(*p. 2308*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.327 src/main/activemq/wireformat/openwire/marshal/v3/SessionInfo File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SessionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SessionInfoMarshaller*** *(p. 2312).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.328 src/main/activemq/wireformat/openwire/marshal/v1/ShutdownI File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::ShutdownInfoMarshaller**
  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(p. 2356).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.329 src/main/activemq/wireformat/openwire/marshal/v2/ShutdownI File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::ShutdownInfoMarshaller**
  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(p. 2352).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.330 src/main/activemq/wireformat/openwire/marshal/v3/ShutdownI File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::ShutdownInfoMarshaller**
  Marshaling code for Open Wire Format for ***ShutdownInfoMarshaller*** *(*p. 2360*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.331 src/main/activemq/wireformat/openwire/marshal/v1/Subscriptio File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::SubscriptionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2496).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.332 src/main/activemq/wireformat/openwire/marshal/v2/Subscriptio File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::SubscriptionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2504).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.333 src/main/activemq/wireformat/openwire/marshal/v3/Subscriptio File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::SubscriptionInfoMarshaller**
  Marshaling code for Open Wire Format for ***SubscriptionInfoMarshaller*** *(p. 2500).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.334 src/main/activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionIdMarshaller**
  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(p. 2577).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.335 src/main/activemq/wireformat/openwire/marshal/v2/TransactionId Marshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(*p. 2585*).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.336 src/main/activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionIdMarshaller**

  Marshaling code for Open Wire Format for ***TransactionIdMarshaller*** *(p. 2581).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.337 src/main/activemq/wireformat/openwire/marshal/v1/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::TransactionInfoMarshaller**
  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2602).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.338 src/main/activemq/wireformat/openwire/marshal/v2/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::TransactionInfoMarshaller**
  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2594).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.339 src/main/activemq/wireformat/openwire/marshal/v3/TransactionInfoMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/BaseCommandMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::TransactionInfoMarshaller**
  Marshaling code for Open Wire Format for ***TransactionInfoMarshaller*** *(p. 2598).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.340 src/main/activemq/wireformat/openwire/marshal/v1/WireForma File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::WireFormatInfoMarshaller**
  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(p. 2717).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.341 src/main/activemq/wireformat/openwire/marshal/v2/WireForma File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::WireFormatInfoMarshaller**
  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(p. 2713).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.342 src/main/activemq/wireformat/openwire/marshal/v3/WireForma File Reference

```
#include <activemq/wireformat/openwire/marshal/BaseDataStreamMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::WireFormatInfoMarshaller**
  Marshaling code for Open Wire Format for ***WireFormatInfoMarshaller*** *(*p. 2709*).*

Namespaces

- namespace **activemq**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.343 src/main/activemq/wireformat/openwire/marshal/v1/XATransac File Reference

```
#include <activemq/wireformat/openwire/marshal/v1/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v1::XATransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2736).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v1**

7.344 src/main/activemq/wireformat/openwire/marshal/v2/XATransactionIdMarshaller.h File Reference

```
#include <activemq/wireformat/openwire/marshal/v2/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v2::XATransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2744).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v2**

7.345 src/main/activemq/wireformat/openwire/marshal/v3/XATransac File Reference

```
#include <activemq/wireformat/openwire/marshal/v3/TransactionIdMarshaller.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <activemq/util/Config.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
```

Data Structures

- class **activemq::wireformat::openwire::marshal::v3::XATransactionIdMarshaller**

  *Marshaling code for Open Wire Format for **XATransactionIdMarshaller** (p. 2740).*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**
- namespace **activemq::wireformat::openwire::marshal::v3**

7.346 src/main/activemq/wireformat/openwire/OpenWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/WireFormatInfo.h>
#include <activemq/commands/DataStructure.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/openwire/utils/BooleanStream.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <memory>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormat**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::marshal**

7.347 src/main/activemq/wireformat/openwire/OpenWireFormatFactor File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/commands/WireFormatInfo.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Pointer.h>
#include <decaf/util/Properties.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatFactory**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.348 src/main/activemq/wireformat/openwire/OpenWireFormatNegot File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <activemq/wireformat/openwire/OpenWireFormat.h>
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/concurrent/CountDownLatch.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/concurrent/atomic/AtomicBoolean.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireFormatNegotiator**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.349 src/main/activemq/wireformat/openwire/OpenWireResponseBuil File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/mock/ResponseBuilder.h>
#include <decaf/util/StlQueue.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::openwire::OpenWireResponseBuilder**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**

7.350 src/main/activemq/wireformat/openwire/utils/BooleanStream.h File Reference

```
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::BooleanStream**

  Manages the writing and reading of boolean data streams to and from a data source such as a DataInputStream or DataOutputStream.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.351 src/main/activemq/wireformat/openwire/utils/HexTable.h File Reference

```
#include <vector>
#include <string>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::HexTable**

  *The **HexTable** (p. 1388) class maps hexidecimal strings to the value of an index into the table, i.e.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.352 src/main/activemq/wireformat/openwire/utils/MessagePropertyI File Reference

```
#include <activemq/util/Config.h>
#include <activemq/commands/Message.h>
#include <activemq/util/PrimitiveMap.h>
#include <activemq/exceptions/ActiveMQException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::MessagePropertyInterceptor**

  Used the base ActiveMQMessage class to intercept calls to get and set properties in order to capture the calls that use the reserved JMS properties and get and set them in the OpenWire Message properties.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.353 src/main/activemq/wireformat/openwire/utils/OpenwireStringSu File Reference

```
#include <string>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::openwire::utils::OpenwireStringSupport**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::openwire**
- namespace **activemq::wireformat::openwire::utils**

7.354 src/main/activemq/wireformat/stomp/StompCommandConstants File Reference

```
#include <cms/Destination.h>
#include <activemq/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <string>
#include <map>
```

Data Structures

- class **activemq::wireformat::stomp::StompCommandConstants**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.355 src/main/activemq/wireformat/stomp/StompFrame.h File Reference

```
#include <string>
#include <string.h>
#include <map>
#include <decaf/util/Properties.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/DataInputStream.h>
#include <activemq/util/Config.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompFrame**

  A Stomp-level message frame that encloses all messages to and from the broker.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.356 src/main/activemq/wireformat/stomp/StompHelper.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/util/LongSequenceGenerator.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/commands/Message.h>
#include <activemq/commands/MessageId.h>
#include <activemq/commands/ProducerId.h>
#include <activemq/commands/ConsumerId.h>
#include <activemq/commands/TransactionId.h>
#include <activemq/commands/ActiveMQDestination.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompHelper**

  Utility Methods used when marshaling to and from StompFrame's.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.357 src/main/activemq/wireformat/stomp/StompWireFormat.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <activemq/wireformat/stomp/StompFrame.h>
#include <activemq/wireformat/stomp/StompHelper.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormat**

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.358 src/main/activemq/wireformat/stomp/StompWireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormatFactory.h>
#include <activemq/wireformat/stomp/StompWireFormat.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::stomp::StompWireFormatFactory**

  Factory used to create the Stomp Wire Format instance.

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**
- namespace **activemq::wireformat::stomp**

7.359 src/main/activemq/wireformat/WireFormat.h File Reference

```
#include <activemq/wireformat/WireFormatNegotiator.h>
#include <decaf/io/DataInputStream.h>
#include <decaf/io/DataOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/lang/Pointer.h>
#include <activemq/util/Config.h>
#include <activemq/commands/Command.h>
#include <activemq/transport/Transport.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **activemq::wireformat::WireFormat**

  *Provides a mechanism to marshal **commands** (p. 59) into and out of packets or into and out of streams, Channels and Datagrams.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.360 src/main/activemq/wireformat/WireFormatFactory.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/wireformat/WireFormat.h>
#include <decaf/util/Properties.h>
#include <decaf/lang/Pointer.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatFactory**

  *The **WireFormatFactory** (p. 2697) is the interface that all **WireFormatFactory** (p. 2697) classes must extend.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.361 src/main/activemq/wireformat/WireFormatNegotiator.h File Reference

```
#include <activemq/util/Config.h>
#include <activemq/transport/TransportFilter.h>
#include <decaf/lang/Pointer.h>
```

Data Structures

- class **activemq::wireformat::WireFormatNegotiator**

  *Defines a **WireFormatNegotiator** (p. 2721) which allows a **WireFormat** (p. 2693) to.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.362 src/main/activemq/wireformat/WireFormatRegistry.h File Reference

```
#include <activemq/util/Config.h>
#include <string>
#include <vector>
#include <activemq/wireformat/WireFormatFactory.h>
#include <decaf/util/StlMap.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **activemq::wireformat::WireFormatRegistry**

  *Registry of all **WireFormat** (p. 2693) Factories that are available to the client at runtime.*

Namespaces

- namespace **activemq**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **activemq::wireformat**

7.363 src/main/cms/BytesMessage.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/Message.h>
```

Data Structures

- class **cms::BytesMessage**

  *A **BytesMessage** (p. 759) object is used to send a message containing a stream of unsigned bytes.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.364 src/main/cms/Closeable.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::Closeable**

  Interface for a class that implements the close method.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.365 src/main/decaf/io/Closeable.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::Closeable**

  Interface for a class that implements the close method.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.366 src/main/cms/CMSException.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <cms/Config.h>
```

Data Structures

- class **cms::CMSException**

  CMS API Exception that is the base for all exceptions thrown from CMS classes.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.367 src/main/cms/CMSProperties.h File Reference

```
#include <cms/Config.h>
#include <map>
#include <string>
#include <vector>
```

Data Structures

- class **cms::CMSProperties**

  Interface for a Java-like properties object.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.368 src/main/cms/CMSSecurityException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::CMSSecurityException**

  This exception must be thrown when a provider rejects a user name/password submitted by a client.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.369 src/main/cms/Connection.h File Reference

```
#include <cms/Config.h>
#include <cms/Startable.h>
#include <cms/Stoppable.h>
#include <cms/Closeable.h>
#include <cms/Session.h>
#include <cms/ConnectionMetaData.h>
```

Data Structures

- class **cms::Connection**

  The client's connection to its provider.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.370 src/main/cms/ConnectionFactory.h File Reference

```
#include <cms/Config.h>
#include <cms/Connection.h>
#include <cms/CMSException.h>
#include <string>
```

Data Structures

- class **cms::ConnectionFactory**

  Defines the interface for a factory that creates connection objects, the ***Connection*** *(p. 941) objects returned implement the CMS* ***Connection*** *(p. 941) interface and hide the CMS Provider specific implementation details behind that interface.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.371 src/main/cms/ConnectionMetaData.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ConnectionMetaData**

  *A **ConnectionMetaData** (p. 1015) object provides information describing the **Connection** (p. 941) object.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.372 src/main/cms/DeliveryMode.h File Reference

`#include <cms/Config.h>`

Data Structures

- class **cms::DeliveryMode**

  This is an Abstract class whose purpose is to provide a container for the delivery mode enumeration for CMS messages.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.373 src/main/cms/Destination.h File Reference

```
#include <cms/CMSProperties.h>
#include <cms/Config.h>
#include <string>
```

Data Structures

- class **cms::Destination**

  *A **Destination** (p. 1190) object encapsulates a provider-specific address.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.374 src/main/cms/ExceptionListener.h File Reference

```
#include <cms/Config.h>
```

```
#include <cms/CMSException.h>
```

Data Structures

- class **cms::ExceptionListener**

  If a CMS provider detects a serious problem, it notifies the client application through an ***ExceptionListener*** *(p. 1275) that is registered with the* ***Connection*** *(p. 941).*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.375 src/main/cms/IllegalStateException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::IllegalStateException**

  This exception is thrown when a method is invoked at an illegal or inappropriate time or if the provider is not in an appropriate state for the requested operation.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.376 src/main/decaf/lang/exceptions/IllegalStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalStateException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.377 src/main/cms/InvalidClientIdException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::InvalidClientIdException**

  This exception must be thrown when a client attempts to set a connection's client ID to a value that is rejected by a provider.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.378 src/main/cms/InvalidDestinationException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::InvalidDestinationException**

  This exception must be thrown when a destination either is not understood by a provider or is no longer valid.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.379 src/main/cms/InvalidSelectorException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::InvalidSelectorException**

  This exception must be thrown when a CMS client attempts to give a provider a message selector with invalid syntax.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.380 src/main/cms/MapMessage.h File Reference

`#include <cms/Config.h>`

`#include <cms/Message.h>`

Data Structures

- class **cms::MapMessage**

  *A **MapMessage** (p. 1701) object is used to send a set of name-value pairs.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.381 src/main/cms/MessageConsumer.h File Reference

```
#include <cms/Config.h>
#include <cms/MessageListener.h>
#include <cms/Message.h>
#include <cms/Closeable.h>
```

Data Structures

- class **cms::MessageConsumer**

  A client uses a `MessageConsumer` (p. 1795) *to received messages from a destination.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.382 src/main/cms/MessageEOFException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::MessageEOFException**

  This exception must be thrown when an unexpected end of stream has been reached when a ***StreamMessage*** *(p. 2476) or* ***BytesMessage*** *(p. 759) is being read.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.383 src/main/cms/MessageFormatException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::MessageFormatException**

  This exception must be thrown when a CMS client attempts to use a data type not supported by a message or attempts to read data in a message as the wrong type.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.384 src/main/cms/MessageListener.h File Reference

```
#include <cms/Config.h>
```

Data Structures

- class **cms::MessageListener**

  A `MessageListener` (p. 1860) object is used to receive asynchronously delivered messages.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.385 src/main/cms/MessageNotReadableException.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::MessageNotReadableException**

  This exception must be thrown when a CMS client attempts to read a write-only message.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.386 src/main/cms/MessageNotWriteableException.h File Reference

```
#include <cms/Config.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::MessageNotWriteableException**

  This exception must be thrown when a CMS client attempts to write to a read-only message.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.387 src/main/cms/MessageProducer.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/Destination.h>
#include <cms/Closeable.h>
#include <cms/CMSException.h>
#include <cms/DeliveryMode.h>
```

Data Structures

- class **cms::MessageProducer**

  A client uses a ***MessageProducer*** (p. *1878*) *object to send messages to a* ***Destination*** (p. *1190*)*.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.388 src/main/cms/ObjectMessage.h File Reference

`#include <cms/Config.h>`

`#include <cms/Message.h>`

Data Structures

- class **cms::ObjectMessage**

  Place holder for interaction with JMS systems that support Java, the C++ client is not responsible for deserializing the contained Object.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.389 src/main/cms/Queue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Queue**

  An interface encapsulating a provider-specific queue name.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.390 src/main/decaf/util/Queue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::util::Queue< E >**

  A kind of collection provides advanced operations than other basic collections, such as insertion, extraction, and inspection.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.391 src/main/cms/QueueBrowser.h File Reference

```
#include <vector>
#include <string>
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Queue.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::QueueBrowser**

  This class implements in interface for browsing the messages in a ***Queue*** *(p. 2157) without removing them.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.392 src/main/cms/Session.h File Reference

```
#include <cms/Config.h>
#include <cms/Closeable.h>
#include <cms/Message.h>
#include <cms/TextMessage.h>
#include <cms/BytesMessage.h>
#include <cms/MapMessage.h>
#include <cms/StreamMessage.h>
#include <cms/MessageProducer.h>
#include <cms/MessageConsumer.h>
#include <cms/Topic.h>
#include <cms/Queue.h>
#include <cms/QueueBrowser.h>
#include <cms/TemporaryTopic.h>
#include <cms/TemporaryQueue.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Session**

  *A **Session** (p. 2270) object is a single-threaded context for producing and consuming messages.*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.393 src/main/cms/Startable.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::Startable**

  Interface for a class that implements the start method.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.394 src/main/cms/Stoppable.h File Reference

`#include <cms/Config.h>`

`#include <cms/CMSException.h>`

Data Structures

- class **cms::Stoppable**

  Interface for a class that implements the stop method.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.395 src/main/cms/StreamMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
#include <cms/MessageNotReadableException.h>
#include <cms/MessageNotWriteableException.h>
#include <cms/MessageFormatException.h>
#include <cms/MessageEOFException.h>
```

Data Structures

- class **cms::StreamMessage**

  Interface for a ***StreamMessage*** *(p. 2476).*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.396 src/main/cms/TemporaryQueue.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryQueue**

  *Defines a Temporary **Queue** (p. 2157) based **Destination** (p. 1190).*

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.397 src/main/cms/TemporaryTopic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TemporaryTopic**

  *Defines a Temporary **Topic*** (p. *2571*) *based **Destination*** (p. *1190*).

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.398 src/main/cms/TextMessage.h File Reference

```
#include <cms/Config.h>
#include <cms/Message.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::TextMessage**

  Interface for a text message.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.399 src/main/cms/Topic.h File Reference

```
#include <cms/Config.h>
#include <cms/Destination.h>
#include <cms/CMSException.h>
```

Data Structures

- class **cms::Topic**

  An interface encapsulating a provider-specific topic name.

Namespaces

- namespace **cms**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

7.400 src/main/decaf/internal/AprPool.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::AprPool**

  Wraps an APR pool object so that classes in ***decaf*** *(p. 94) can create a static member for use in static methods where apr function calls that need a pool are made.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

7.401 src/main/decaf/internal/DecafRuntime.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runtime.h>
#include <apr_pools.h>
```

Data Structures

- class **decaf::internal::DecafRuntime**

  Handles APR initialization and termination.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**

7.402 src/main/decaf/internal/io/StandardErrorOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardErrorOutputStream**

  Wrapper Around the Standard error Output facility on the current platform.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.403 src/main/decaf/internal/io/StandardInputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardInputStream**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.404 src/main/decaf/internal/io/StandardOutputStream.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::internal::io::StandardOutputStream**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::io**

7.405 src/main/decaf/internal/net/URIEncoderDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <string>
```

Data Structures

- class **decaf::internal::net::URIEncoderDecoder**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.406 src/main/decaf/internal/net/URIHelper.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/internal/net/URIType.h>
```

Data Structures

- class **decaf::internal::net::URIHelper**

  Helper class used by the URI classes in encoding and decoding of URI's.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.407 src/main/decaf/internal/net/URIType.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::internal::net::URIType**

  Basic type object that holds data that composes a given URI.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::net**

7.408 src/main/decaf/internal/nio/BufferFactory.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/LongBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/ShortBuffer.h>
```

Data Structures

- class **decaf::internal::nio::BufferFactory**

  *Factory class used by static methods in the **decaf::nio** (p. 106) package to create the various default version of the NIO interfaces.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.409 src/main/decaf/internal/nio/ByteArrayBuffer.h File Reference

```
#include <decaf/nio/ByteBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
#include <decaf/nio/CharBuffer.h>
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/nio/FloatBuffer.h>
#include <decaf/nio/ShortBuffer.h>
#include <decaf/nio/IntBuffer.h>
#include <decaf/nio/LongBuffer.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayBuffer**

  This class defines six categories of operations upon byte buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.410 src/main/decaf/internal/nio/ByteArrayPerspective.h File Reference

```
#include <decaf/internal/util/ByteArrayAdapter.h>
```

Data Structures

- class **decaf::internal::nio::ByteArrayPerspective**

  This class extends ByteArray to create a reference counted byte array that can be held and used by several different ByteBuffers and allow them to know on destruction whose job it is to delete the perspective.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.411 src/main/decaf/internal/nio/CharArrayBuffer.h File Reference

```
#include <decaf/nio/CharBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::CharArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.412 src/main/decaf/internal/nio/DoubleArrayBuffer.h File Reference

```
#include <decaf/nio/DoubleBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::DoubleArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.413 src/main/decaf/internal/nio/FloatArrayBuffer.h File Reference

```
#include <decaf/nio/FloatBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::FloatArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.414 src/main/decaf/internal/nio/IntArrayBuffer.h File Reference

```
#include <decaf/nio/IntBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::IntArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.415 src/main/decaf/internal/nio/LongArrayBuffer.h File Reference

```
#include <decaf/nio/LongBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::LongArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.416 src/main/decaf/internal/nio/ShortArrayBuffer.h File Reference

```
#include <decaf/nio/ShortBuffer.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::internal::nio::ShortArrayBuffer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::nio**

7.417 src/main/decaf/internal/util/ByteArrayAdapter.h File Reference

```
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
```

Data Structures

- class **decaf::internal::util::ByteArrayAdapter**

  This class adapts primitive type arrays to a base byte array so that the classes can inter-operate on the same base byte array without copying data.

- union **decaf::internal::util::ByteArrayAdapter::Array**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.418 src/main/decaf/internal/util/HexStringParser.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::internal::util::HexStringParser**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::internal**
- namespace **decaf::internal::util**

7.419 src/main/decaf/io/BlockingByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::BlockingByteArrayInputStream**

  This is a blocking version of a byte buffer stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.420 src/main/decaf/io/BufferedInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
```

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::io::BufferedInputStream**

  A wrapper around another input stream that performs a buffered read, where it reads more data than it needs in order to reduce the number of ***io*** *(*p. 100*) operations on the input stream.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.421 src/main/decaf/io/BufferedOutputStream.h File Reference

`#include <decaf/io/FilterOutputStream.h>`

`#include <decaf/lang/exceptions/IllegalArgumentException.h>`

Data Structures

- class **decaf::io::BufferedOutputStream**

  Wrapper around another output stream that buffers output before writing to the target output stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.422 src/main/decaf/io/ByteArrayInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
#include <algorithm>
```

Data Structures

- class **decaf::io::ByteArrayInputStream**

  Simple implementation of ***InputStream*** *(p. 1406) that wraps around an STL Vector std::vector<unsigned char>.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.423 src/main/decaf/io/ByteArrayOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/util/concurrent/Mutex.h>
#include <vector>
```

Data Structures

- class **decaf::io::ByteArrayOutputStream**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.424 src/main/decaf/io/DataInputStream.h File Reference

```
#include <decaf/io/FilterInputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/EOFException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::DataInputStream**

  A data input stream lets an application read primitive Java data types from an underlying input stream in a machine-independent way.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.425 src/main/decaf/io/DataOutputStream.h File Reference

```
#include <decaf/io/FilterOutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/io/UTFDataFormatException.h>
#include <string>
```

Data Structures

- class **decaf::io::DataOutputStream**

  A data output stream lets an application write primitive Java data types to an output stream in a portable way.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.426 src/main/decaf/io/EOFException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::EOFException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.427 src/main/decaf/io/FilterInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
```

Data Structures

- class **decaf::io::FilterInputStream**

  *A **FilterInputStream** (p. 1315) contains some other input stream, which it uses as its basic source of data, possibly transforming the data along the way or providing additional functionality.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.428 src/main/decaf/io/FilterOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::FilterOutputStream**

  This class is the superclass of all classes that filter output streams.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.429 src/main/decaf/io/InputStream.h File Reference

```
#include <decaf/io/IOException.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::io::InputStream**

  Base interface for an input stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.430 src/main/decaf/io/InterruptedIOException.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::InterruptedIOException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.431 src/main/decaf/io/IOException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::io::IOException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.432 src/main/decaf/io/OutputStream.h File Reference

```
#include <decaf/io/Closeable.h>
#include <decaf/io/IOException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::OutputStream**

  Base interface for an output stream.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.433 src/main/decaf/io/Reader.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/InputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Reader**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.434 src/main/decaf/io/UTFDataFormatException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::io::UTFDataFormatException**

  Thrown from classes that attempt to read or write a UTF-8 encoded string and an encoding error is encountered.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.435 src/main/decaf/io/Writer.h File Reference

```
#include <string>
#include <decaf/io/IOException.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::io::Writer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::io**

7.436 src/main/decaf/lang/Appendable.h File Reference

```
#include <decaf/lang/Exception.h>
```

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Appendable**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.437 src/main/decaf/lang/Boolean.h File Reference

```
#include <string>
#include <decaf/lang/Comparable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Boolean**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.438 src/main/decaf/lang/Byte.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Byte**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.439 src/main/decaf/lang/Character.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
```

Data Structures

- class **decaf::lang::Character**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.440 src/main/decaf/lang/CharSequence.h File Reference

`#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>`

`#include <decaf/util/Config.h>`

Data Structures

- class **decaf::lang::CharSequence**

  *A **CharSequence** (p. 833) is a readable sequence of char values.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.441 src/main/decaf/lang/Comparable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Comparable< T >**

  This interface imposes a total ordering on the objects of each class that implements it.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.442 src/main/decaf/lang/Double.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Double**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.443 src/main/decaf/lang/Exception.h File Reference

```
#include <decaf/lang/Throwable.h>
#include <decaf/lang/exceptions/ExceptionDefines.h>
#include <decaf/util/Config.h>
#include <stdarg.h>
#include <sstream>
```

Data Structures

- class **decaf::lang::Exception**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.444 src/main/decaf/lang/exceptions/ClassCastException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::ClassCastException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.445 src/main/decaf/lang/exceptions/IllegalArgumentException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalArgumentException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.446 src/main/decaf/lang/exceptions/IllegalMonitorStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IllegalMonitorStateException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.447 src/main/decaf/lang/exceptions/IndexOutOfBoundsException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::IndexOutOfBoundsException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.448 src/main/decaf/lang/exceptions/InterruptedException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InterruptedException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.449 src/main/decaf/lang/exceptions/InvalidStateException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::InvalidStateException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.450 src/main/decaf/lang/exceptions/NoSuchElementException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NoSuchElementException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.451 src/main/decaf/lang/exceptions/NullPointerException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NullPointerException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.452 src/main/decaf/lang/exceptions/NumberFormatException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::NumberFormatException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.453 src/main/decaf/lang/exceptions/RuntimeException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::RuntimeException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.454 src/main/decaf/lang/exceptions/UnsupportedOperationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::lang::exceptions::UnsupportedOperationException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **decaf::lang::exceptions**

7.455 src/main/decaf/lang/Float.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Float**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.456 src/main/decaf/lang/Integer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <string>
#include <decaf/lang/exceptions/NumberFormatException.h>
```

Data Structures

- class **decaf::lang::Integer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.457 src/main/decaf/lang/Iterable.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/util/Iterator.h>
```

Data Structures

- class **decaf::lang::Iterable< E >**

  *Implementing this interface allows an object to be cast to an **Iterable** (p. 1487) type for generic collections API calls.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.458 src/main/decaf/lang/Long.h File Reference

```
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Long**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.459 src/main/decaf/lang/Math.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Math**

  The class `Math` *(p. 1718) contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.460 src/main/decaf/lang/Number.h File Reference

`#include <decaf/util/Config.h>`

Data Structures

- class **decaf::lang::Number**

  *The abstract class **Number** (p. 1954) is the superclass of classes **Byte** (p. 664), **Double** (p. 1231), **Float** (p. 1328), **Integer** (p. 1428), **Long** (p. 1652), and **Short** (p. 2321).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.461 src/main/decaf/lang/Pointer.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/ClassCastException.h>
#include <decaf/util/concurrent/atomic/AtomicInteger.h>
#include <decaf/util/Comparator.h>
#include <memory>
#include <algorithm>
```

Data Structures

- class **decaf::lang::AtomicRefCounter**
- struct **decaf::lang::STATIC_CAST_TOKEN**
- struct **decaf::lang::DYNAMIC_CAST_TOKEN**
- class **decaf::lang::Pointer< T, REFCOUNTER >**

  *Decaf's implementation of a Smart **Pointer** (p. 2011) that is a template on a Type and is **Thread** (p. 2544) Safe if the default Reference Counter is used.*

- class **decaf::lang::PointerComparator< T, R >**

  *This implementation of Comparator is designed to allows objects in a Collection to be sorted or tested for equality based on the value of the Object being Pointed to and not the value of the contained pointer in the **Pointer** (p. 2011) instance.*

- struct **std::less< decaf::lang::Pointer< T > >**

  An override of the less function object so that the Pointer objects can be stored in STL Maps, etc.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**
- namespace **std**

Functions

- template<typename T , typename R , typename U >
  bool **decaf::lang::operator==** (const Pointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator==** (const U *left, const Pointer< T, R > &right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator!=** (const Pointer< T, R > &left, const U *right)
- template<typename T , typename R , typename U >
  bool **decaf::lang::operator!=** (const U *left, const Pointer< T, R > &right)

7.462 src/main/decaf/lang/Runnable.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runnable**

  Interface for a runnable object - defines a task that can be run by a thread.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.463 src/main/decaf/lang/Runtime.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Runtime**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.464 src/main/decaf/lang/Short.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NumberFormatException.h>
#include <string>
```

Data Structures

- class **decaf::lang::Short**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.465 src/main/decaf/lang/System.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/Exception.h>
#include <decaf/internal/AprPool.h>
#include <string>
```

Data Structures

- class **decaf::lang::System**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.466 src/main/decaf/lang/Thread.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/lang/Runnable.h>
#include <decaf/util/Config.h>
#include <stdexcept>
#include <assert.h>
#include <apr_thread_proc.h>
```

Data Structures

- class **decaf::lang::Thread**

  Basic thread class - mimics the Java ***Thread*** *(p. 2544).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.467 src/main/decaf/lang/Throwable.h File Reference

```
#include <string>
#include <vector>
#include <iostream>
#include <exception>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::lang::Throwable**

  This class represents an error that has occurred.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::lang**

7.468 src/main/decaf/net/BindException.h File Reference

```
#include <decaf/util/Config.h>
```

```
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::BindException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.469 src/main/decaf/net/BufferedSocket.h File Reference

```
#include <decaf/net/Socket.h>
#include <decaf/net/SocketException.h>
#include <decaf/io/BufferedInputStream.h>
#include <decaf/io/BufferedOutputStream.h>
```

Data Structures

- class **decaf::net::BufferedSocket**

  Buffered ***Socket*** *(p. 2371) class that wraps a* `Socket` *(p. 2371) derived object and provides Buffered input and Output Streams to improce the efficiency of the reads and writes.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.470 src/main/decaf/net/ConnectException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::ConnectException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.471 src/main/decaf/net/HttpRetryException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::HttpRetryException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.472 src/main/decaf/net/MalformedURLException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::MalformedURLException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.473 src/main/decaf/net/NoRouteToHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::NoRouteToHostException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.474 src/main/decaf/net/PortUnreachableException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/net/SocketException.h>
```

Data Structures

- class **decaf::net::PortUnreachableException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.475 src/main/decaf/net/ProtocolException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::ProtocolException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.476 src/main/decaf/net/ServerSocket.h File Reference

```
#include <decaf/net/TcpSocket.h>
#include <decaf/net/SocketException.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::ServerSocket**

  A server socket class (for testing purposes).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.477 src/main/decaf/net/Socket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Closeable.h>
#include <decaf/util/Config.h>
#include <apr_network_io.h>
```

Data Structures

- class **decaf::net::Socket**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.478 src/main/decaf/net/SocketError.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketError**

  Static utility class to simplify handling of error codes for socket operations.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.479 src/main/decaf/net/SocketException.h File Reference

```
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::SocketException**

  Exception for errors when manipulating sockets.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.480 src/main/decaf/net/SocketFactory.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/util/Properties.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::net::SocketFactory**

  Socket *(p. 2371) Factory implementation for use in Creating Sockets.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.481 src/main/decaf/net/SocketInputStream.h File Reference

```
#include <decaf/io/InputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
#include <decaf/lang/exceptions/NullPointerException.h>
```

Data Structures

- class **decaf::net::SocketInputStream**

  Input stream for performing reads on a socket.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.482 src/main/decaf/net/SocketOutputStream.h File Reference

```
#include <decaf/io/OutputStream.h>
#include <decaf/net/Socket.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::net::SocketOutputStream**

  Output stream for performing write operations on a socket.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.483 src/main/decaf/net/SocketTimeoutException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/InterruptedIOException.h>
```

Data Structures

- class **decaf::net::SocketTimeoutException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.484 src/main/decaf/net/TcpSocket.h File Reference

```
#include <decaf/net/SocketException.h>
#include <decaf/net/Socket.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
```

Data Structures

- class **decaf::net::TcpSocket**

  Platform-independent implementation of the socket interface.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.485 src/main/decaf/net/UnknownHostException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownHostException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.486 src/main/decaf/net/UnknownServiceException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::net::UnknownServiceException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.487 src/main/decaf/net/URI.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/net/URISyntaxException.h>
#include <decaf/net/MalformedURLException.h>
#include <decaf/net/URL.h>
#include <decaf/internal/net/URIType.h>
#include <string>
```

Data Structures

- class **decaf::net::URI**

  *This class represents an instance of a **URI** (p. 2638) as defined by RFC 2396.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.488 src/main/decaf/net/URISyntaxException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::net::URISyntaxException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.489 src/main/decaf/net/URL.h File Reference

`#include <decaf/util/Config.h>`

`#include <string>`

Data Structures

- class **decaf::net::URL**

  *Class **URL** (p. 2677) represents a Uniform Resource Locator, a pointer to a "resource" on the World Wide Web.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.490 src/main/decaf/net/URLDecoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLDecoder**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.491 src/main/decaf/net/URLEncoder.h File Reference

```
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::net::URLEncoder**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::net**

7.492 src/main/decaf/nio/Buffer.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

```
#include <decaf/nio/InvalidMarkException.h>
```

Data Structures

- class **decaf::nio::Buffer**

  A container for data of a specific primitive type.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.493 src/main/decaf/nio/BufferOverflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferOverflowException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.494 src/main/decaf/nio/BufferUnderflowException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::nio::BufferUnderflowException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.495 src/main/decaf/nio/ByteBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/internal/nio/ByteArrayPerspective.h>
```

Data Structures

- class **decaf::nio::ByteBuffer**

  This class defines six categories of operations upon byte buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.496 src/main/decaf/nio/CharBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
#include <decaf/lang/CharSequence.h>
#include <decaf/lang/Appendable.h>
```

Data Structures

- class **decaf::nio::CharBuffer**

  This class defines four categories of operations upon character buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.497 src/main/decaf/nio/DoubleBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::DoubleBuffer**

  This class defines four categories of operations upon double buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.498 src/main/decaf/nio/FloatBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::FloatBuffer**

  This class defines four categories of operations upon float buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.499 src/main/decaf/nio/IntBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::IntBuffer**

  This class defines four categories of operations upon int buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.500 src/main/decaf/nio/InvalidMarkException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::nio::InvalidMarkException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.501 src/main/decaf/nio/LongBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::LongBuffer**

  This class defines four categories of operations upon long long buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.502 src/main/decaf/nio/ReadOnlyBufferException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::nio::ReadOnlyBufferException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.503 src/main/decaf/nio/ShortBuffer.h File Reference

```
#include <decaf/nio/Buffer.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/nio/BufferUnderflowException.h>
#include <decaf/nio/BufferOverflowException.h>
#include <decaf/nio/ReadOnlyBufferException.h>
```

Data Structures

- class **decaf::nio::ShortBuffer**

  This class defines four categories of operations upon short buffers:.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::nio**

7.504 src/main/decaf/security/auth/x500/X500Principal.h File Reference

```
#include <string>
#include <vector>
#include <decaf/security/Principal.h>
#include <decaf/util/Map.h>
#include <decaf/io/InputStream.h>
```

Data Structures

- class **decaf::security::auth::x500::X500Principal**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::auth**
- namespace **decaf::security::auth::x500**

7.505 src/main/decaf/security/cert/Certificate.h File Reference

```
#include <vector>
#include <decaf/util/Config.h>
#include <decaf/security/InvalidKeyException.h>
#include <decaf/security/NoSuchAlgorithmException.h>
#include <decaf/security/SignatureException.h>
#include <decaf/security/cert/CertificateEncodingException.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::Certificate**

  Base interface for all identity certificates.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.506 src/main/decaf/security/cert/CertificateEncodingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateEncodingException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.507 src/main/decaf/security/cert/CertificateException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::cert::CertificateException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.508 src/main/decaf/security/cert/CertificateExpiredException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateExpiredException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.509 src/main/decaf/security/cert/CertificateNotYetValidException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateNotYetValidException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.510 src/main/decaf/security/cert/CertificateParsingException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/cert/CertificateException.h>
```

Data Structures

- class **decaf::security::cert::CertificateParsingException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.511 src/main/decaf/security/cert/X509Certificate.h File Reference

```
#include <decaf/security/cert/Certificate.h>
#include <decaf/util/Config.h>
#include <decaf/util/Date.h>
```

Data Structures

- class **decaf::security::cert::X509Certificate**

  Base interface for all identity certificates.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**
- namespace **decaf::security::cert**

7.512 src/main/decaf/security/GeneralSecurityException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security::GeneralSecurityException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.513 src/main/decaf/security/InvalidKeyException.h File Reference

```
#include <decaf/security/KeyException.h>
```

Data Structures

- class **decaf::security::InvalidKeyException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.514 src/main/decaf/security/Key.h File Reference

```
#include <vector>
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Key**

  *The **Key** (p. 1570) interface is the top-level interface for all keys.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.515 src/main/decaf/security/KeyException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::KeyException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.516 src/main/decaf/security/NoSuchAlgorithmException.h File Reference

```
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchAlgorithmException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.517 src/main/decaf/security/NoSuchProviderException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::NoSuchProviderException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.518 src/main/decaf/security/Principal.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::Principal**

  Base interface for a principal, which can represent an individual or organization.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.519 src/main/decaf/security/PublicKey.h File Reference

```
#include <decaf/security/Key.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::security::PublicKey**

  A public key.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.520 src/main/decaf/security/SignatureException.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/security/GeneralSecurityException.h>
```

Data Structures

- class **decaf::security::SignatureException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security**

7.521 src/main/decaf/security_provider/SecurityProvider.h File Reference

Data Structures

- class **decaf::security_provider::SecurityProvider**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.522 src/main/decaf/security_provider/SecurityProviderMap.h File Reference

```
#include <map>
#include <vector>
#include <string>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderMap**

  Lookup Map for Connector Factories.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.523 src/main/decaf/security_provider/SecurityProviderRegistrar.h File Reference

```
#include <string>
#include <decaf/security_provider/SecurityProviderMap.h>
```

Data Structures

- class **decaf::security_provider::SecurityProviderRegistrar**

  Registers the passed in provider into the provider map, this class can manage the lifetime of the registered provider (default behaviour).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**

7.524 src/main/decaf/security_provider/unix/openssl/OpenSSLX500Pr File Reference

```
#include <decaf/security/auth/x500/X500Principal.h>
#include <openssl/x509.h>
```

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX500Principal**

  The `OpenSSLX500Principal` (p. 1961) wraps around an OpenSSL `X509_NAME` structure.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**
- namespace **decaf::security_provider::unix**
- namespace **decaf::security_provider::unix::openssl**

7.525 src/main/decaf/security_provider/unix/openssl/OpenSSLX509C File Reference

`#include <decaf/security/cert/X509Certificate.h>`

Data Structures

- class **decaf::security_provider::unix::openssl::OpenSSLX509Certificate**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::security_provider**
- namespace **decaf::security_provider::unix**
- namespace **decaf::security_provider::unix::openssl**

7.526 src/main/decaf/util/AbstractCollection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Collection.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractCollection< E >**

  This class provides a skeletal implementation of the ***Collection*** *(p. 871) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.527 src/main/decaf/util/AbstractList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/List.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractList< E >**

  *This class provides a skeletal implementation of the **List** (p. 1591) interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.528 src/main/decaf/util/AbstractMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Map.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractMap< K, V, COMPARATOR >**

  *This class provides a skeletal implementation of the **Map** (p. 1689) interface, to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.529 src/main/decaf/util/AbstractQueue.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Queue.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractQueue< E >**

  *This class provides skeletal implementations of some **Queue** (p. 2154) operations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.530 src/main/decaf/util/AbstractSequentialList.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractList.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSequentialList< E >**

  This class provides a skeletal implementation of the ***List*** *(p. 1591) interface to minimize the effort required to implement this interface backed by a "sequential access" data store (such as a linked list).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.531 src/main/decaf/util/AbstractSet.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/Set.h>
#include <memory>
```

Data Structures

- class **decaf::util::AbstractSet< E >**

  *This class provides a skeletal implementation of the **Set** (p. 2320) interface to minimize the effort required to implement this interface.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.532 src/main/decaf/util/Collection.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/Iterable.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Synchronizable.h>
```

Data Structures

- class **decaf::util::Collection< E >**

  The root interface in the collection hierarchy.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.533 src/main/decaf/util/Comparator.h File Reference

```
#include <decaf/util/Config.h>
#include <algorithm>
#include <functional>
```

Data Structures

- class **decaf::util::Comparator< T >**

  A comparison function, which imposes a total ordering on some collection of objects.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.534 src/main/decaf/util/concurrent/atomic/AtomicBoolean.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicBoolean**

  A boolean value that may be updated atomically.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.535 src/main/decaf/util/concurrent/atomic/AtomicInteger.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Number.h>
#include <string>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicInteger**

  An int value that may be updated atomically.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.536 src/main/decaf/util/concurrent/atomic/AtomicReference.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Long.h>
#include <apr_atomic.h>
```

Data Structures

- class **decaf::util::concurrent::atomic::AtomicReference< T >**

  An Pointer reference that may be updated atomically.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::atomic**

7.537 src/main/decaf/util/concurrent/BrokenBarrierException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::BrokenBarrierException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.538 src/main/decaf/util/concurrent/Callable.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::Callable< V >**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.539 src/main/decaf/util/concurrent/CancellationException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CancellationException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.540 src/main/decaf/util/concurrent/Concurrent.h File Reference

```
#include <decaf/util/concurrent/Lock.h>
```

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

Defines

- #define **WAIT_INFINITE** 0xFFFFFFFF

  The synchronized macro defines a mechanism for snycronizing a section of code.

- #define **synchronized**(W)

7.540.1 Define Documentation

7.540.1.1 #define synchronized(W)

Value:

```
if(false){}                                                    \
        else                                                           \
        for( decaf::util::concurrent::Lock lock_W(W);                  \
             lock_W.isLocked(); lock_W.unlock() )
```

7.540.1.2 #define WAIT_INFINITE 0xFFFFFFFF

The synchronized macro defines a mechanism for snycronizing a section of code. The macro must be passed an object that implements the Syncronizable interface.

The macro works by creating a for loop that will loop exactly once, creating a Lock object that is scoped to the loop. Once the loop completes and exits the Lock object goes out of scope releasing the lock on object W. For added safety the if else is used because not all compiles restrict the lifetime of loop variables to the loop, they will however restrict them to the scope of the else.

The macro would be used as follows.

Syncronizable X;

somefunction() { syncronized(X) { // Do something that needs syncronizing. } }

7.541 src/main/decaf/util/concurrent/ConcurrentMap.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/util/Map.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentMap< K, V, COMPARATOR >**

  *Interface for a **Map** (p. 1689) type that provides additional **atomic** (p. 118) putIfAbsent, remove, and replace methods alongside the already available **Map** (p. 1689) interface.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.542 src/main/decaf/util/concurrent/ConcurrentStlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/ConcurrentMap.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::concurrent::ConcurrentStlMap< K, V, COMPARATOR >**

  Map *(p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.543 src/main/decaf/util/concurrent/CountDownLatch.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::CountDownLatch**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.544 src/main/decaf/util/concurrent/Delayed.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/TimeUnit.h>
```

Data Structures

- class **decaf::util::concurrent::Delayed**

  A mix-in style interface for marking objects that should be acted upon after a given delay.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.545 src/main/decaf/util/concurrent/ExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::ExecutionException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.546 src/main/decaf/util/concurrent/Executor.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Runnable.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/util/concurrent/RejectedExecutionException.h>
```

Data Structures

- class **decaf::util::concurrent::Executor**

  An object that executes submitted ***decaf.lang.Runnable*** *(p. 2256) tasks.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.547 src/main/decaf/util/concurrent/Future.h File Reference

Data Structures

- class **decaf::util::concurrent::Future< V >**

  *A **Future** (p. 1374) represents the result of an asynchronous computation.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.548 src/main/decaf/util/concurrent/Lock.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Lock**

  A wrapper class around a given synchronization mechanism that provides automatic release upon destruction.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.549 src/main/decaf/util/concurrent/locks/Lock.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/util/concurrent/locks/Condition.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Lock**

  Lock *(p. 1618) implementations provide more extensive locking operations than can be obtained using synchronized statements.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.550 src/main/decaf/util/concurrent/locks/Condition.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/InterruptedException.h>
#include <decaf/lang/exceptions/IllegalMonitorStateException.h>
```

Data Structures

- class **decaf::util::concurrent::locks::Condition**

  Condition (p. *932*) *factors out the* ***Mutex*** (p. *1921*) *monitor methods (wait, notify and notifyAll) into distinct objects to give the effect of having multiple wait-sets per object, by combining them with the use of arbitrary* ***Lock*** (p. *1618*) *implementations.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.551 src/main/decaf/util/concurrent/locks/ReadWriteLock.h File Reference

Data Structures

- class **decaf::util::concurrent::locks::ReadWriteLock**

  *A **ReadWriteLock** (p. 2170) maintains a pair of associated **locks** (p. 119), one for read-only operations and one for writing.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**
- namespace **decaf::util::concurrent::locks**

7.552 src/main/decaf/util/concurrent/Mutex.h File Reference

```
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/lang/Thread.h>
#include <decaf/util/Config.h>
#include <decaf/internal/AprPool.h>
#include <apr_thread_mutex.h>
#include <apr_thread_cond.h>
#include <list>
#include <assert.h>
```

Data Structures

- class **decaf::util::concurrent::Mutex**

  Creates a pthread_ mutex_ t object.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.553 src/main/decaf/util/concurrent/PooledThread.h File Reference

```
#include <decaf/lang/Thread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThread**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.554 src/main/decaf/util/concurrent/PooledThreadListener.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::PooledThreadListener**

  Abstract Listener Interface for users of `ThreadPool` *(p. 2549).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.555 src/main/decaf/util/concurrent/RejectedExecutionException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::RejectedExecutionException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.556 src/main/decaf/util/concurrent/Synchronizable.h File Reference

```
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::Synchronizable**

  The interface for all synchronizable objects (that is, objects that can be locked and unlocked).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.557 src/main/decaf/util/concurrent/TaskListener.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/lang/Exception.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::concurrent::TaskListener**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.558 src/main/decaf/util/concurrent/ThreadFactory.h File Reference

Data Structures

- class **decaf::util::concurrent::ThreadFactory**
  public interface ***ThreadFactory*** *(p. 2547)*

Namespaces

- namespace **decaf**
  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.
- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.559 src/main/decaf/util/concurrent/ThreadPool.h File Reference

```
#include <decaf/lang/Runnable.h>
#include <decaf/util/concurrent/PooledThread.h>
#include <decaf/util/concurrent/PooledThreadListener.h>
#include <decaf/util/concurrent/TaskListener.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/StlQueue.h>
#include <decaf/util/logging/LoggerDefines.h>
#include <decaf/util/Config.h>
#include <vector>
```

Data Structures

- class **decaf::util::concurrent::ThreadPool**

  Defines a Thread Pool object that implements the functionality of pooling threads to perform user tasks.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.560 src/main/decaf/util/concurrent/TimeoutException.h File Reference

```
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::concurrent::TimeoutException**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.561 src/main/decaf/util/concurrent/TimeUnit.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::concurrent::TimeUnit**

  *A **TimeUnit** (p. 2562) represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::concurrent**

7.562 src/main/decaf/util/Date.h File Reference

```
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::Date**

  Wrapper class around a time value in milliseconds.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.563 src/main/decaf/util/Iterator.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IllegalStateException.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
```

Data Structures

- class **decaf::util::Iterator< T >**

  Defines an object that can be used to iterate over the elements of a collection.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.564 src/main/decaf/util/List.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
#include <decaf/util/ListIterator.h>
```

Data Structures

- class **decaf::util::List< E >**

  An ordered collection (also known as a sequence).

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.565 src/main/decaf/util/ListIterator.h File Reference

```
#include <decaf/util/Iterator.h>
#include <decaf/util/Config.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
```

Data Structures

- class **decaf::util::ListIterator< E >**

  An iterator for lists that allows the programmer to traverse the list in either direction, modify the list during iteration, and obtain the iterator's current position in the list.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.566 src/main/decaf/util/logging/ConsoleHandler.h File Reference

```
#include <decaf/util/logging/StreamHandler.h>
#include <decaf/io/StandardErrorOutputStream.h>
```

7.567 src/main/decaf/util/logging/Filter.h File Reference

`#include <decaf/util/logging/LogRecord.h>`

Data Structures

- class **decaf::util::logging::Filter**

  *A **Filter** (p. 1314) can be used to provide fine grain control over what is logged, beyond the control provided by log levels.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.568 src/main/decaf/util/logging/Formatter.h File Reference

Data Structures

- class **decaf::util::logging::Formatter**

  *A **Formatter** (p. 1372) provides support for formatting LogRecords.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.569 src/main/decaf/util/logging/Handler.h File Reference

`#include <decaf/io/Closeable.h>`

`#include <decaf/util/logging/LogRecord.h>`

Data Structures

- class **decaf::util::logging::Handler**

  *A **Handler** (p. 1382) object takes log messages from a **Logger** (p. 1623) and exports them.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.570 src/main/decaf/util/logging/Logger.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/LogRecord.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <list>
#include <string>
#include <stdarg.h>
```

Data Structures

- class **decaf::util::logging::Logger**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.571 src/main/decaf/util/logging/LoggerCommon.h File Reference

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

Enumerations

- enum **decaf::util::logging::Level** {

  decaf::util::logging::Off, **decaf::util::logging::Null**, **decaf::util::logging::Markblock**, **decaf::util::logging::Debug**,

  decaf::util::logging::Info, **decaf::util::logging::Warn**, **decaf::util::logging::Error**, **decaf::util::logging::Fatal**,

  decaf::util::logging::Throwing }

  Defines an enumeration for logging levels.

7.572 src/main/decaf/util/logging/LoggerDefines.h File Reference

```
#include <decaf/util/logging/SimpleLogger.h>
#include <sstream>
```

Defines

- #define **LOGDECAF_DECLARE**(loggerName) static **decaf::util::logging::SimpleLogger** loggerName;
- #define **LOGDECAF_INITIALIZE**(loggerName, className, loggerFamily) **decaf::util::logging::SimpleLogger** className::loggerName(loggerFamily);
- #define **LOGDECAF_DECLARE_LOCAL**(loggerName) **decaf::util::logging::Logger** loggerName;
- #define **LOGDECAF_DEBUG**(logger, message) logger.debug(__FILE__, __LINE__, message);
- #define **LOGDECAF_DEBUG_1**(logger, message, value)
- #define **LOGDECAF_INFO**(logger, message) logger.info(__FILE__, __LINE__, message);
- #define **LOGDECAF_ERROR**(logger, message) logger.error(__FILE__, __LINE__, message);
- #define **LOGDECAF_WARN**(logger, message) logger.warn(__FILE__, __LINE__, message);
- #define **LOGDECAF_FATAL**(logger, message) logger.fatal(__FILE__, __LINE__, message);

7.572.1 Define Documentation

7.572.1.1 #define LOGDECAF_DEBUG(logger, message) logger.debug(__FILE__, __LINE__, message);

7.572.1.2 #define LOGDECAF_DEBUG_1(logger, message, value)

Value:

```
;                   \
   {                                                    \
      std::ostringstream ostream;                       \
      ostream << message << value;                      \
      logger.debug(__FILE__, __LINE__, ostream.str());  \
   }
```

7.572.1.3 #define LOGDECAF_DECLARE(loggerName) static decaf::util::logging::SimpleLogger loggerName;

7.572.1.4 #define LOGDECAF_DECLARE_LOCAL(loggerName) decaf::util::logging::Logger loggerName;

7.572.1.5 #define LOGDECAF_ERROR(logger, message) logger.error(__FILE__, __LINE__, message);

7.572.1.6 #define LOGDECAF_FATAL(logger, message) logger.fatal(__FILE__, __LINE__, message);

7.572.1.7 #define LOGDECAF_INFO(logger, message) logger.info(__FILE__, __LINE__, message);

7.572.1.8 #define LOGDECAF_INITIALIZE(loggerName, className, loggerFamily) decaf::util::logging::SimpleLogger className::loggerName(loggerFamily);

7.572.1.9 #define LOGDECAF_WARN(logger, message) logger.warn(__FILE__, __LINE__, message);

7.573 src/main/decaf/util/logging/LoggerHierarchy.h File Reference

Data Structures

- class **decaf::util::logging::LoggerHierarchy**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.574 src/main/decaf/util/logging/LogManager.h File Reference

```
#include <map>
#include <list>
#include <string>
#include <vector>
#include <decaf/util/Properties.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::LogManager**

  *There is a single global **LogManager** (p. 1640) object that is used to maintain a set of shared state about Loggers and log services.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.575 src/main/decaf/util/logging/LogRecord.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::logging::LogRecord**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.576 src/main/decaf/util/logging/LogWriter.h File Reference

```
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::logging::LogWriter**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.577 src/main/decaf/util/logging/MarkBlockLogger.h File Reference

```
#include <decaf/util/logging/Logger.h>
```

Data Structures

- class **decaf::util::logging::MarkBlockLogger**

  Defines a class that can be used to mark the entry and exit from scoped blocks.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.578 src/main/decaf/util/logging/PropertiesChangeListener.h File Reference

`#include <decaf/util/Config.h>`

Data Structures

- class **decaf::util::logging::PropertiesChangeListener**

  Defines the interface that classes can use to listen for change events on ***Properties*** *(p. 2140).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.579 src/main/decaf/util/logging/SimpleFormatter.h File Reference

`#include <decaf/util/logging/formatter.h>`

Data Structures

- class **decaf::util::logging::SimpleFormatter**

  *Print a brief summary of the **LogRecord** (p. 1645) in a human readable format.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.580 src/main/decaf/util/logging/SimpleLogger.h File Reference

```
#include <string>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::SimpleLogger**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.581 src/main/decaf/util/logging/StreamHandler.h File Reference

```
#include <decaf/util/logging/LoggerCommon.h>
#include <decaf/util/logging/Handler.h>
#include <decaf/util/logging/Formatter.h>
#include <decaf/util/logging/Filter.h>
#include <decaf/io/OutputStream.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/lang/exceptions/InvalidStateException.h>
#include <decaf/util/concurrent/Concurrent.h>
#include <decaf/util/Config.h>
```

Data Structures

- class **decaf::util::logging::StreamHandler**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**
- namespace **decaf::util::logging**

7.582 src/main/decaf/util/Map.h File Reference

```
#include <vector>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
```

Data Structures

- class **decaf::util::Map< K, V, COMPARATOR >**

  Map *(p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

- class **decaf::util::Map< K, V, COMPARATOR >::Entry**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.583 src/main/decaf/util/Properties.h File Reference

```
#include <memory>
#include <vector>
#include <string>
#include <decaf/util/Config.h>
#include <decaf/io/InputStream.h>
#include <decaf/io/OutputStream.h>
#include <decaf/io/Reader.h>
#include <decaf/io/Writer.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/lang/exceptions/NullPointerException.h>
#include <decaf/io/IOException.h>
```

Data Structures

- class **decaf::util::Properties**

  Java-like properties class for mapping string names to string values.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.584 src/main/decaf/util/Random.h File Reference

```
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <decaf/util/Config.h>
#include <vector>
#include <cmath>
```

Data Structures

- class **decaf::util::Random**

  Random (p. *2160*) *Value Generator which is used to generate a stream of pseudorandom numbers.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.585 src/main/decaf/util/Set.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractCollection.h>
```

Data Structures

- class **decaf::util::Set< E >**

  A collection that contains no duplicate elements.

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.586 src/main/decaf/util/StlList.h File Reference

```
#include <list>
#include <algorithm>
#include <memory>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/lang/exceptions/IndexOutOfBoundsException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Config.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/ListIterator.h>
#include <decaf/util/List.h>
```

Data Structures

- class **decaf::util::StlList< E >**

  List (p. *1591*) *class that wraps the STL list object to provide a simpler interface and additional methods not provided by the STL type.*

- class **decaf::util::StlList< E >::StlListIterator**
- class **decaf::util::StlList< E >::ConstStlListIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.587 src/main/decaf/util/StlMap.h File Reference

```
#include <map>
#include <vector>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Map.h>
```

Data Structures

- class **decaf::util::StlMap< K, V, COMPARATOR >**

  Map *(p. 1689) template that wraps around a std::map to provide a more user-friendly interface and to provide common functions that do not exist in std::map.*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.588 src/main/decaf/util/StlQueue.h File Reference

```
#include <list>
#include <vector>
#include <decaf/util/Iterator.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/lang/Exception.h>
```

Data Structures

- class **decaf::util::StlQueue< T >**

  *The **Queue** (p. 2154) class accepts messages with an psuh(m) command where m is the message to be queued.*

- class **decaf::util::StlQueue< T >::QueueIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.589 src/main/decaf/util/StlSet.h File Reference

```
#include <set>
#include <vector>
#include <memory>
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/concurrent/Synchronizable.h>
#include <decaf/util/concurrent/Mutex.h>
#include <decaf/util/Iterator.h>
#include <decaf/util/AbstractSet.h>
```

Data Structures

- class **decaf::util::StlSet< E >**

  Set *(p. 2320) template that wraps around a std::set to provide a more user-friendly interface and to provide common functions that do not exist in std::set.*

- class **decaf::util::StlSet< E >::SetIterator**
- class **decaf::util::StlSet< E >::ConstSetIterator**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.590 src/main/decaf/util/StringTokenizer.h File Reference

```
#include <decaf/lang/exceptions/NoSuchElementException.h>
#include <decaf/util/Config.h>
#include <string>
```

Data Structures

- class **decaf::util::StringTokenizer**

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

7.591 src/main/decaf/util/UUID.h File Reference

```
#include <decaf/util/Config.h>
#include <decaf/lang/Comparable.h>
#include <decaf/lang/exceptions/UnsupportedOperationException.h>
#include <decaf/lang/exceptions/IllegalArgumentException.h>
#include <apr_uuid.h>
#include <string>
```

Data Structures

- class **decaf::util::UUID**

  *A class that represents an immutable universally unique identifier (**UUID** (p. 2686)).*

Namespaces

- namespace **decaf**

  Licensed to the Apache Software Foundation (ASF) under one or more contributor license agreements.

- namespace **decaf::util**

Index