StdAir

1.00.2

Generated by Doxygen 1.6.1

Sat Jun 13 18:44:04 2015

Contents

1 StdAir Documentation

1.1 Getting Started

- Main features
- Installation
- Linking with StdAir
- Users Guide
- Tutorials
- Copyright and License
- Make a Difference
- Make a new release
- People

1.2 StdAir at SourceForge

- `Project page`
- `Download StdAir`
- `Open a ticket fora bug or feature`

- Mailing lists
- Forums
  - Discuss about Development issues
  - Ask for Help
  - Discuss StdAir

1.3 StdAir Development

- Git Repository (Subversion is deprecated)
- Coding Rules
- Documentation Rules
- Test Rules

1.4 External Libraries

- Boost (C++ STL extensions)
- ZeroMQ (networking made easy)
- Python
- MySQL client
- SOCI (C++ DB API)

1.5 Support StdAir

1.6 About StdAir

StdAir is a C++ library of classes and functions modeling typical airline IT business objects. For instance, it is used by the C++ Revenue Management Open Library project (http://sourceforge.net/projects/rmol/). StdAir mainly targets simulation purposes. N

StdAir makes an extensive use of existing open-source libraries for increased functionality, speed and accuracy. In particular Boost (*C++ STL Extensions*) library is used.

The StdAir library originates from the department of Operational Research and Innovation at Amadeus, Sophia Antipolis, France. StdAir is released under the terms of the GNU Lesser General Public License (LGPLv2.1) for you to enjoy.

StdAir should work on GNU/Linux, Sun Solaris, Microsoft Windows (with Cygwin, MinGW/MSYS, or Microsoft Visual C++ .NET) and Mac OS X operating systems.

Note:

(N) - The StdAir library is **NOT** intended, in any way, to be used by airlines for production systems. If you want to report issue, bug or feature request, or if you just want to give feedback, have a look on the right-hand side of this page for the preferred reporting methods. In any case, please do not contact Amadeus directly for any matter related to StdAir.

2 BomAbstract

Abstract part of the Business Object Model (BOM)

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010

3 C++ Utility Class Browsing and Dumping the StdAir BOM Tree

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <ostream>
// StdAir
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/OnDDate.hpp>

namespace stdair {

  struct FlagSaver {
  public:
    FlagSaver (std::ostream& oStream)
      : _oStream (oStream), _streamFlags (oStream.flags()) {
    }

    ~FlagSaver() {
      // Reset formatting flags of the given output stream
      _oStream.flags (_streamFlags);
    }

  private:
    std::ostream& _oStream;
    std::ios::fmtflags _streamFlags;
  };
```

```
// ////////////////////////////////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const BomRoot& iBomRoot,
                       const AirlineCode_T& iAirlineCode,
                       const FlightNumber_T& iFlightNumber) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are Inventory objects
  if (BomManager::hasList<Inventory> (iBomRoot) == false) {
    return;
  }

  // Browse the inventories
  unsigned short invIdx = 1;
  const InventoryList_T& lInventoryList =
    BomManager::getList<Inventory> (iBomRoot);
  for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
       itInv != lInventoryList.end(); ++itInv, ++invIdx) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Retrieve the inventory key (airline code)
    const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();

    // Display only the requested inventories
    if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
      // Get the list of flight-dates for that inventory
      list (oStream, *lInv_ptr, invIdx, iFlightNumber);
    }
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
                       const unsigned short iInventoryIndex,
                       const FlightNumber_T& iFlightNumber) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are FlightDate objects
  if (BomManager::hasMap<FlightDate> (iInventory) == false) {
    return;
  }

  //
  const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
  oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;

  // Browse the flight-dates
  unsigned short lCurrentFlightNumber = 0;
  unsigned short flightNumberIdx = 0;
  unsigned short departureDateIdx = 1;
  const FlightDateMap_T& lFlightDateList =
    BomManager::getMap<FlightDate> (iInventory);
  for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
       itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
    const FlightDate* lFD_ptr = itFD->second;
    assert (lFD_ptr != NULL);

    // Retrieve the key of the flight-date
    const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
    const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

    // Display only the requested flight number
    if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
      //
```

```
      if (lCurrentFlightNumber != lFlightNumber) {
        lCurrentFlightNumber = lFlightNumber;
        ++flightNumberIdx; departureDateIdx = 1;
        oStream << "  " << iInventoryIndex << "." << flightNumberIdx << ". "
                << lAirlineCode << lFlightNumber << std::endl;
      }

      oStream << "    " << iInventoryIndex << "." << flightNumberIdx
              << "." << departureDateIdx << ". "
              << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
              << std::endl;
    }
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
                                           const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are AirportPair objects
  if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
    return;
  }

  const AirportPairList_T& lAirportPairList =
    BomManager::getList<AirportPair> (iBomRoot);
  for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
       itAir != lAirportPairList.end(); ++itAir ) {
    const AirportPair* lAir_ptr = *itAir;
    assert (lAir_ptr != NULL);

    // Check whether there are date-period objects
    assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);

    // Browse the date-period objects
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (*lAir_ptr);

    for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
         itDP != lDatePeriodList.end(); ++itDP) {
      const DatePeriod* lDP_ptr = *itDP;
      assert (lDP_ptr != NULL);

      // Display the date-period object
      oStream << lAir_ptr->describeKey()
              <<" / " << lDP_ptr->describeKey() << std::endl;
    }

  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                             const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << std::endl;
  oStream << "================================================================"
          << std::endl;
  oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
  oStream << "================================================================"
          << std::endl;

  // Check whether there are Inventory objects
```

```
  if (BomManager::hasList<Inventory> (iBomRoot) == false) {
    return;
  }

  // Browse the inventories
  const InventoryList_T& lInventoryList =
    BomManager::getList<Inventory> (iBomRoot);
  for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
       itInv != lInventoryList.end(); ++itInv) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Display the inventory
    csvDisplay (oStream, *lInv_ptr);
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                             const Inventory& iInventory) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
  oStream << "Inventory: " << iInventory.describeKey() << std::endl;
  oStream << "++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;

  // Check whether there are FlightDate objects
  if (BomManager::hasList<FlightDate> (iInventory) == false) {
    return;
  }

  // Browse the flight-dates
  const FlightDateList_T& lFlightDateList =
    BomManager::getList<FlightDate> (iInventory);
  for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
       itFD != lFlightDateList.end(); ++itFD) {
    const FlightDate* lFD_ptr = *itFD;
    assert (lFD_ptr != NULL);

    // Display the flight-date
    csvDisplay (oStream, *lFD_ptr);
  }

  // Check if the inventory contains a list of partners

  if (BomManager::hasList<Inventory> (iInventory)){

    // Browse the partner's inventories
    const InventoryList_T& lPartnerInventoryList =
      BomManager::getList<Inventory> (iInventory);

    for (InventoryList_T::const_iterator itInv = lPartnerInventoryList.begin();

         itInv != lPartnerInventoryList.end(); ++itInv) {

      oStream << "--------------------------------------------------" << std::en
    dl;
      oStream << "Partner inventory:" << std::endl;
      oStream << "--------------------------------------------------" << std::en
    dl;
      const Inventory* lInv_ptr = *itInv;
      assert (lInv_ptr != NULL);

      // Display the inventory
      csvDisplay (oStream, *lInv_ptr);
    }
```

```
      oStream << "*****************************************" << std::endl;
      oStream << std::endl;
    }

    // Check if the inventory contains a list of O&D dates

    if (BomManager::hasList<OnDDate> (iInventory)){

      //Browse the O&Ds
      const OnDDateList_T& lOnDDateList =
        BomManager::getList<OnDDate> (iInventory);

      for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
           itOnD != lOnDDateList.end(); ++itOnD) {
        oStream << "*****************************************" << std::endl;
        oStream << "O&D-Date:" << std::endl;
        oStream << "----------" << std::endl;
        oStream << "Airline, Date, Origin-Destination, Segments, " << std::endl;

        const OnDDate* lOnDDate_ptr = *itOnD;
        assert (lOnDDate_ptr != NULL);

        // Display the O&D date
        csvDisplay (oStream, *lOnDDate_ptr);
      }
      oStream << "*****************************************" << std::endl;
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDisplay (std::ostream& oStream,
                               const OnDDate& iOnDDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    const AirlineCode_T& lAirlineCode = iOnDDate.getAirlineCode();
    const Date_T& lDate = iOnDDate.getDate();
    const AirportCode_T& lOrigin = iOnDDate.getOrigin();
    const AirportCode_T& lDestination = iOnDDate.getDestination();

    oStream << lAirlineCode <<", " << lDate << ", "<< lOrigin << "-"
            << lDestination << ", " << iOnDDate.describeKey() << ", "
            << std::endl;

    const StringDemandStructMap_T& lDemandInfoMap =
      iOnDDate.getDemandInfoMap();

    // Check if the map contains information.
    const bool isInfoMapEmpty = lDemandInfoMap.empty();
    if (isInfoMapEmpty) {
      return;
    }
    assert (lDemandInfoMap.empty() ==false);

    oStream << "----------" << std::endl;
    oStream << "Cabin-Class path, Demand mean, Demand std dev, Yield, "
            << std::endl;

    for (StringDemandStructMap_T::const_iterator itDI = lDemandInfoMap.begin();
         itDI != lDemandInfoMap.end(); ++itDI) {

      const std::string& lCabinClassPath = itDI->first;
      const YieldDemandPair_T lYieldDemandPair =
        itDI->second;
      const Yield_T lYield = lYieldDemandPair.first;
      const MeanStdDevPair_T lMeanStdDevPair =
        lYieldDemandPair.second;
```

```
      const MeanValue_T lDemandMean = lMeanStdDevPair.first;
      const StdDevValue_T lDemandStdDev = lMeanStdDevPair.second;

      oStream << lCabinClassPath << ", "
              << lDemandMean << ", "
              << lDemandStdDev << ", "
              << lYield << ", "
              << std::endl;
    }

  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDisplay (std::ostream& oStream,
                               const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    oStream << "*****************************************" << std::endl;
    oStream << "FlightDate: " << lAirlineCode << iFlightDate.describeKey()
            << std::endl;
    oStream << "*****************************************" << std::endl;

    //
    csvSegmentDateDisplay (oStream, iFlightDate);
    //
    csvLegDateDisplay (oStream, iFlightDate);

    //
    csvLegCabinDisplay (oStream, iFlightDate);

    //
    csvBucketDisplay (oStream, iFlightDate);

    //
    csvFareFamilyDisplay (oStream, iFlightDate);

    //
    csvBookingClassDisplay (oStream, iFlightDate);
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvLegDateDisplay (std::ostream& oStream,
                                      const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "Leg-Dates:" << std::endl
            << "----------" << std::endl;
    oStream << "Flight, Leg, BoardDate, BoardTime, "
            << "OffDate, OffTime, Date Offset, Time Offset, Elapsed, "
            << "Distance, Capacity, " << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
      return;
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
```

```
    BomManager::getList<LegDate> (iFlightDate);
  for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
       itLD != lLegDateList.end(); ++itLD) {
    const LegDate* lLD_ptr = *itLD;
    assert (lLD_ptr != NULL);

    oStream << lAirlineCode << lFlightNumber << " "
            << lFlightDateDate << ", ";

    oStream << lLD_ptr->getBoardingPoint() << "-"
            << lLD_ptr->getOffPoint() << ", "
            << lLD_ptr->getBoardingDate() << ", "
            << lLD_ptr->getBoardingTime() << ", "
            << lLD_ptr->getOffDate() << ", "
            << lLD_ptr->getOffTime() << ", "
            << lLD_ptr->getElapsedTime() << ", "
            << lLD_ptr->getDateOffset().days() << ", "
            << lLD_ptr->getTimeOffset() << ", "
            << lLD_ptr->getDistance() << ", "
            << lLD_ptr->getCapacity() << ", " << std::endl;
  }
  oStream << "*****************************************" << std::endl;
}

// ///////////////////////////////////////////////////////////////////////
void BomDisplay::csvSegmentDateDisplay (std::ostream& oStream,
                                        const FlightDate& iFlightDate) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "*****************************************" << std::endl;
  oStream << "SegmentDates:" << std::endl
          << "----------" << std::endl;
  oStream << "Flight, Segment, Date"
          << std::endl;

  // Retrieve the key of the flight-date
  const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
  const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
  const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

  // Check whether there are SegmentDate objects
  if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
    return;
  }

  // Browse the segment-dates
  const SegmentDateList_T& lSegmentDateList =
    BomManager::getList<SegmentDate> (iFlightDate);
  for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
       itSD != lSegmentDateList.end(); ++itSD) {
    const SegmentDate* lSD_ptr = *itSD;
    assert (lSD_ptr != NULL);

    // Retrieve the key of the segment-date, as well as its dates
    const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
    const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
    const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
    oStream << lAirlineCode << lFlightNumber << " " << lFlightDateDate << ", "
            << lBoardPoint << "-" << lOffPoint << ", " << lSegmentDateDate << s
    td::endl;

    // Check if the current segment has corresponding marketing segments.
    const bool hasMarketingSDList = BomManager::hasList<SegmentDate>(*lSD_ptr);

    if (hasMarketingSDList == true) {
      //
```

```
      const SegmentDateList_T& lMarketingSDList = BomManager::getList<SegmentDa
    te>(*lSD_ptr);

      oStream << " *** Marketed by ";
      for (SegmentDateList_T::const_iterator itMarketingSD = lMarketingSDList.b
    egin();
           itMarketingSD != lMarketingSDList.end(); ++itMarketingSD) {
        SegmentDate* lMarketingSD_ptr = *itMarketingSD;
        FlightDate* lMarketingFD_ptr = BomManager::getParentPtr<FlightDate>(*lM
    arketingSD_ptr);
        Inventory* lMarketingInv_ptr = BomManager::getParentPtr<Inventory>(*lMa
    rketingFD_ptr);
        oStream << lMarketingInv_ptr->toString() << lMarketingFD_ptr->toString(
    ) <<" * ";
      }
    }

    // Check if the current segment is operated by another segment date.
    const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate ();
    if (lOperatingSD_ptr != NULL) {

      const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate>
    (*lOperatingSD_ptr);
      const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory>(
    *lOperatingFD_ptr);
      oStream << " *** Operated by " << lOperatingInv_ptr->toString()
              << lOperatingFD_ptr->toString() << std::endl;
    }

    oStream << std::endl;
  }
}

// ////////////////////////////////////////////////////////////////////////
void BomDisplay::csvLegCabinDisplay (std::ostream& oStream,
                                     const FlightDate& iFlightDate) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "*****************************************" << std::endl;
  oStream << "LegCabins:" << std::endl
          << "----------" << std::endl;
  oStream << "Flight, Leg, Cabin, "
          << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
          << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
          << std::endl;

  // Retrieve the key of the flight-date
  const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
  const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
  const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

  // Check whether there are LegDate objects
  if (BomManager::hasList<LegDate> (iFlightDate) == false) {
    return;
  }

  // Browse the leg-dates
  const LegDateList_T& lLegDateList =
    BomManager::getList<LegDate> (iFlightDate);
  for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
       itLD != lLegDateList.end(); ++itLD) {
    const LegDate* lLD_ptr = *itLD;
    assert (lLD_ptr != NULL);

    // Retrieve the key of the leg-date, as well as its off point
    const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
```

```
      const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

      // Browse the leg-cabins
      const LegCabinList_T& lLegCabinList =
        BomManager::getList<LegCabin> (*lLD_ptr);
      for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
           itLC != lLegCabinList.end(); ++itLC) {
        const LegCabin* lLC_ptr = *itLC;
        assert (lLC_ptr != NULL);

        oStream << lAirlineCode << lFlightNumber << " "
                << lFlightDateDate << ", ";

        oStream << lBoardPoint << "-" << lOffPoint
                << " " << lLegDateDate << ", ";

        oStream << lLC_ptr->getCabinCode() << ", ";

        oStream << lLC_ptr->getOfferedCapacity() << ", "
                << lLC_ptr->getPhysicalCapacity() << ", "
                << lLC_ptr->getRegradeAdjustment() << ", "
                << lLC_ptr->getAuthorizationLevel() << ", "
                << lLC_ptr->getUPR() << ", "
                << lLC_ptr->getSoldSeat() << ", "
                << lLC_ptr->getStaffNbOfSeats() << ", "
                << lLC_ptr->getWLNbOfSeats() << ", "
                << lLC_ptr->getGroupNbOfSeats() << ", "
                << lLC_ptr->getCommittedSpace() << ", "
                << lLC_ptr->getAvailabilityPool() << ", "
                << lLC_ptr->getAvailability() << ", "
                << lLC_ptr->getNetAvailability() << ", "
                << lLC_ptr->getGrossAvailability() << ", "
                << lLC_ptr->getAvgCancellationPercentage() << ", "
                << lLC_ptr->getETB() << ", "
                << lLC_ptr->getCurrentBidPrice() << ", "
                << std::endl;
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
                                           const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvFareFamilyDisplay (std::ostream& oStream,
                                         const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "SegmentCabins:" << std::endl
            << "--------------" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, "
            << "CommSpace, AvPool, BP, " << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
```

```
    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

      // Retrieve the key of the segment-date, as well as its dates
      const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

      // Browse the segment-cabins
      const SegmentCabinList_T& lSegmentCabinList =
        BomManager::getList<SegmentCabin> (*lSD_ptr);
      for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
           itSC != lSegmentCabinList.end(); ++itSC) {
        const SegmentCabin* lSC_ptr = *itSC;
        assert (lSC_ptr != NULL);

        // Retrieve the key of the segment-cabin
        const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

        // Check whether there are fare family objects
        if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
          continue;
        }

        // Browse the fare families
        const FareFamilyList_T& lFareFamilyList =
          BomManager::getList<FareFamily> (*lSC_ptr);
        for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
             itFF != lFareFamilyList.end(); ++itFF) {
          const FareFamily* lFF_ptr = *itFF;
          assert (lFF_ptr != NULL);

          oStream << lAirlineCode << lFlightNumber << " "
                  << lFlightDateDate << ", ";

          oStream << lBoardPoint << "-" << lOffPoint << " "
                  << lSegmentDateDate << ", ";

          oStream << lCabinCode << ", " << lFF_ptr->getFamilyCode() << ", ";

          oStream << lSC_ptr->getBookingCounter() << ", "
                  << lSC_ptr->getMIN() << ", "
                  << lSC_ptr->getUPR() << ", "
                  << lSC_ptr->getCommittedSpace() << ", "
                  << lSC_ptr->getAvailabilityPool() << ", "
                  << lSC_ptr->getCurrentBidPrice() << ", "
                  << std::endl;
        }
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBucketDisplay (std::ostream& oStream,
                                     const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
```

```
FlagSaver flagSaver (oStream);

oStream << "*****************************************" << std::endl;
oStream << "Buckets:" << std::endl
        << "--------" << std::endl;
oStream << "Flight, Leg, Cabin, Yield, AU/SI, SS, AV, "
        << std::endl;

// Retrieve the key of the flight-date
const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

// Check whether there are LegDate objects
if (BomManager::hasList<LegDate> (iFlightDate) == false) {
  return;
}

// Browse the leg-dates
const LegDateList_T& lLegDateList =
  BomManager::getList<LegDate> (iFlightDate);
for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
     itLD != lLegDateList.end(); ++itLD) {
  const LegDate* lLD_ptr = *itLD;
  assert (lLD_ptr != NULL);

  // Retrieve the key of the leg-date, as well as its off point
  const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
  const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
  const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

  // Browse the leg-cabins
  const LegCabinList_T& lLegCabinList =
    BomManager::getList<LegCabin> (*lLD_ptr);
  for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
       itLC != lLegCabinList.end(); ++itLC) {
    const LegCabin* lLC_ptr = *itLC;
    assert (lLC_ptr != NULL);

    // Check whether there are bucket objects
    if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
      continue;
    }

    // Retrieve the key of the leg-cabin
    const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();

    // Browse the buckets
    const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);

    for (BucketList_T::const_iterator itBuck = lBucketList.begin();
         itBuck != lBucketList.end(); ++itBuck) {
      const Bucket* lBucket_ptr = *itBuck;
      assert (lBucket_ptr != NULL);

      oStream << lAirlineCode << lFlightNumber << " "
              << lFlightDateDate << ", ";

      oStream << lBoardPoint << "-" << lOffPoint << " "
              << lLegDateDate << ", " << lCabinCode << ", ";

      oStream << lBucket_ptr->getYieldRangeUpperValue() << ", "
              << lBucket_ptr->getSeatIndex() << ", "
              << lBucket_ptr->getSoldSeats() << ", "
              << lBucket_ptr->getAvailability() << ", ";
      oStream << std::endl;
    }
```

```
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
                                           const BookingClass& iBookingClass,
                                           const std::string& iLeadingString) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << iLeadingString << iBookingClass.getClassCode();

    if (iBookingClass.getSubclassCode() == 0) {
      oStream << ", ";
    } else {
      oStream << iBookingClass.getSubclassCode() << ", ";
    }
    oStream << iBookingClass.getAuthorizationLevel() << " ("
            << iBookingClass.getProtection() << "), "
            << iBookingClass.getNegotiatedSpace() << ", "
            << iBookingClass.getNoShowPercentage() << ", "
            << iBookingClass.getCancellationPercentage() << ", "
            << iBookingClass.getNbOfBookings() << ", "
            << iBookingClass.getNbOfGroupBookings() << " ("
            << iBookingClass.getNbOfPendingGroupBookings() << "), "
            << iBookingClass.getNbOfStaffBookings() << ", "
            << iBookingClass.getNbOfWLBookings() << ", "
            << iBookingClass.getETB() << ", "
            << iBookingClass.getNetClassAvailability() << ", "
            << iBookingClass.getNetRevenueAvailability() << ", "
            << iBookingClass.getSegmentAvailability() << ", "
            << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
                                           const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Headers
    oStream << "*****************************************" << std::endl;
    oStream << "Subclasses:" << std::endl
            << "-----------" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), "
            << "Nego, NS%, OB%, "
            << "Bkgs, GrpBks (pdg), StfBkgs, WLBkgs, ETB, "
            << "ClassAvl, RevAvl, SegAvl, "
            << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
```

```
const SegmentDate* lSD_ptr = *itSD;
assert (lSD_ptr != NULL);

// Retrieve the key of the segment-date, as well as its dates
const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

// Browse the segment-cabins
const SegmentCabinList_T& lSegmentCabinList =
  BomManager::getList<SegmentCabin> (*lSD_ptr);
for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
     itSC != lSegmentCabinList.end(); ++itSC) {
  const SegmentCabin* lSC_ptr = *itSC;
  assert (lSC_ptr != NULL);

  // Retrieve the key of the segment-cabin
  const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

  // Build the leading string to be displayed
  std::ostringstream oSCLeadingStr;
  oSCLeadingStr << lAirlineCode << lFlightNumber << " "
                << lFlightDateDate << ", "
                << lBoardPoint << "-" << lOffPoint << " "
                << lSegmentDateDate << ", "
                << lCabinCode << ", ";

  // Default Fare Family code, when there are no FF
  FamilyCode_T lFamilyCode ("NoFF");

  // Check whether there are FareFamily objects
  if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {

    // Browse the fare families
    const FareFamilyList_T& lFareFamilyList =
      BomManager::getList<FareFamily> (*lSC_ptr);
    for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
         itFF != lFareFamilyList.end(); ++itFF) {
      const FareFamily* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Retrieve the key of the segment-cabin
      lFamilyCode = lFF_ptr->getFamilyCode();

      // Complete the leading string to be displayed
      std::ostringstream oFFLeadingStr;
      oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

      // Browse the booking-classes
      const BookingClassList_T& lBookingClassList =
        BomManager::getList<BookingClass> (*lFF_ptr);
      for (BookingClassList_T::const_iterator itBC =
             lBookingClassList.begin();
           itBC != lBookingClassList.end(); ++itBC) {
        const BookingClass* lBC_ptr = *itBC;
        assert (lBC_ptr != NULL);

        //
        csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
      }
    }

    // Go on to the next segment-cabin
    continue;
  }
  assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);
```

```
      // The fare family code is a fake one ('NoFF'), and therefore
      // does not vary
      std::ostringstream oFFLeadingStr;
      oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

      // Browse the booking-classes, directly from the segment-cabin object
      const BookingClassList_T& lBookingClassList =
        BomManager::getList<BookingClass> (*lSC_ptr);
      for (BookingClassList_T::const_iterator itBC =
             lBookingClassList.begin();
           itBC != lBookingClassList.end(); ++itBC) {
        const BookingClass* lBC_ptr = *itBC;
        assert (lBC_ptr != NULL);

        //
        csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
      }
    }
  }
  oStream << "*****************************************" << std::endl;
}

// ///////////////////////////////////////////////////////////////////
void BomDisplay::
csvDisplay (std::ostream& oStream,
            const TravelSolutionList_T& iTravelSolutionList) {

  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "Travel solutions:";
  unsigned short idx = 0;
  for (TravelSolutionList_T::const_iterator itTS =
         iTravelSolutionList.begin();
       itTS != iTravelSolutionList.end(); ++itTS, ++idx) {
    const TravelSolutionStruct& lTS = *itTS;

    oStream << std::endl;
    oStream << "    [" << idx << "] " << lTS.display();
  }
}

// ///////////////////////////////////////////////////////////////////
void BomDisplay::
csvDisplay (std::ostream& oStream,
            const DatePeriodList_T& iDatePeriodList) {

  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Browse the date-period objects
  for (DatePeriodList_T::const_iterator itDP = iDatePeriodList.begin();
       itDP != iDatePeriodList.end(); ++itDP) {
    const DatePeriod* lDP_ptr = *itDP;
    assert (lDP_ptr != NULL);

    // Display the date-period object
    csvDateDisplay (oStream, *lDP_ptr);
  }
}

// ///////////////////////////////////////////////////////////////////
void BomDisplay::csvSimFQTAirRACDisplay (std::ostream& oStream,
                                         const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);
```

```
  oStream << std::endl;
  oStream << "==============================================================="
          << std::endl;
  oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
  oStream << "==============================================================="
          << std::endl;

  // Check whether there are airport-pair objects
  if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
    return;
  }

  // Browse the airport-pair objects
  const AirportPairList_T& lAirportPairList =
    BomManager::getList<AirportPair> (iBomRoot);
  for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
       itAir != lAirportPairList.end(); ++itAir ) {
    const AirportPair* lAir_ptr = *itAir;
    assert (lAir_ptr != NULL);

    // Display the airport pair object
    csvAirportPairDisplay (oStream, *lAir_ptr);
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvAirportPairDisplay (std::ostream& oStream,
                                        const AirportPair& iAirportPair) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "+++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
  oStream << "AirportPair: " << iAirportPair.describeKey() << std::endl;
  oStream << "+++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;

  // Check whether there are date-period objects
  if (BomManager::hasList<DatePeriod> (iAirportPair) == false) {
    return;
  }

  // Browse the date-period objects
  const DatePeriodList_T& lDatePeriodList =
    BomManager::getList<DatePeriod> (iAirportPair);
  for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
       itDP != lDatePeriodList.end(); ++itDP) {
    const DatePeriod* lDP_ptr = *itDP;
    assert (lDP_ptr != NULL);

    // Display the date-period object
    csvDateDisplay (oStream, *lDP_ptr);
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDateDisplay (std::ostream& oStream,
                                 const DatePeriod& iDatePeriod) {

  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "-----------------------------------------" << std::endl;
  oStream << "DatePeriod: " << iDatePeriod.describeKey() << std::endl;
  oStream << "-----------------------------------------" << std::endl;

  // Check whether there are pos-channel objects
  if (BomManager::hasList<PosChannel> (iDatePeriod) == false) {
    return;
```

```
    }

    // Browse the pos-channel objects
    const PosChannelList_T& lPosChannelList =
      BomManager::getList<PosChannel> (iDatePeriod);
    for (PosChannelList_T::const_iterator itPC = lPosChannelList.begin();
         itPC != lPosChannelList.end(); ++itPC) {
      const PosChannel* lPC_ptr = *itPC;
      assert (lPC_ptr != NULL);

      // Display the pos-channel object
      csvPosChannelDisplay (oStream, *lPC_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvPosChannelDisplay (std::ostream& oStream,
                                         const PosChannel& iPosChannel) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "PosChannel: " << iPosChannel.describeKey() << std::endl;
    oStream << "*****************************************" << std::endl;

    // Check whether there are time-period objects
    if (BomManager::hasList<TimePeriod> (iPosChannel) == false) {
      return;
    }

    // Browse the time-period objects
    const TimePeriodList_T& lTimePeriodList =
      BomManager::getList<TimePeriod> (iPosChannel);
    for (TimePeriodList_T::const_iterator itTP = lTimePeriodList.begin();
         itTP != lTimePeriodList.end(); ++itTP) {
      const TimePeriod* lTP_ptr = *itTP;
      assert (lTP_ptr != NULL);

      // Display the time-period object
      csvTimeDisplay (oStream, *lTP_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvTimeDisplay (std::ostream& oStream,
                                   const TimePeriod& iTimePeriod) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "----------------------------------------" << std::endl;
    oStream << "TimePeriod: " << iTimePeriod.describeKey() << std::endl;
    oStream << "----------------------------------------" << std::endl;

    // Only one of the fare/yield feature list exists. Each of the following
    // two methods will check for the existence of the list. So, only the
    // existing list will be actually displayed.
    csvFeatureListDisplay<FareFeatures> (oStream, iTimePeriod);
    csvFeatureListDisplay<YieldFeatures> (oStream, iTimePeriod);
  }

  // ////////////////////////////////////////////////////////////////////
  template <typename FEATURE_TYPE>
  void BomDisplay::csvFeatureListDisplay (std::ostream& oStream,
                                          const TimePeriod& iTimePeriod) {

    // Check whether there are fare/yield-feature objects
```

```
    if (BomManager::hasList<FEATURE_TYPE> (iTimePeriod) == false) {
      return;
    }

    // Browse the fare/yield-feature objects
    typedef typename BomHolder<FEATURE_TYPE>::BomList_T FeaturesList_T;
    const FeaturesList_T& lFeaturesList =
      BomManager::getList<FEATURE_TYPE> (iTimePeriod);
    for (typename FeaturesList_T::const_iterator itFF = lFeaturesList.begin();
         itFF != lFeaturesList.end(); ++itFF) {
      const FEATURE_TYPE* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Display the fare-features object
      csvFeaturesDisplay (oStream, *lFF_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  template <typename FEATURE_TYPE>
  void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
                                       const FEATURE_TYPE& iFeatures) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "--------------------------------------" << std::endl;
    oStream << "Fare/yield-Features: " << iFeatures.describeKey() << std::endl;
    oStream << "--------------------------------------" << std::endl;

    // Check whether there are airlineClassList objects
    if (BomManager::hasList<AirlineClassList> (iFeatures) == false) {
      return;
    }

    // Browse the airlineClassList objects
    const AirlineClassListList_T& lAirlineClassListList =
      BomManager::getList<AirlineClassList> (iFeatures);
    for (AirlineClassListList_T::const_iterator itACL =
           lAirlineClassListList.begin();
         itACL != lAirlineClassListList.end(); ++itACL) {
      const AirlineClassList* lACL_ptr = *itACL;
      assert (lACL_ptr != NULL);

      // Display the airlineClassList object
      csvAirlineClassDisplay(oStream, *lACL_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::
  csvAirlineClassDisplay  (std::ostream& oStream,
                           const AirlineClassList& iAirlineClassList) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "-----------------------------------" << std::endl;
    oStream << "AirlineClassList: "
            << iAirlineClassList.describeKey() << std::endl;
    oStream << "-----------------------------------" << std::endl;
  }

}

/*!
```

4 KeyAbstract

Part of the Business Object Model (BOM) handling (hash-like)keys

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010

5 C++ Class Building Sample StdAir BOM Trees

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/FlightPeriod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {

    // DEBUG
    STDAIR_LOG_DEBUG ("StdAir is building the BOM tree from built-in "
                      << "specifications.");
```

```
  // ////// Basic Bom Tree ///////
  // Build the inventory (flight-dates) and the schedule (flight period)
  // parts.
  buildSampleInventorySchedule (ioBomRoot);

  // Build the pricing (fare rules) and revenue accounting (yields) parts.
  buildSamplePricing (ioBomRoot);

  // ////// Partnership Bom Tree ///////
  // Build the inventory (flight-dates) and the schedule (flight period)
  // parts.
  buildPartnershipsSampleInventoryAndRM (ioBomRoot);

  // Build the pricing (fare rules) and revenue accounting (yields) parts.
  buildPartnershipsSamplePricing (ioBomRoot);

  // Build a dummy inventory, needed by RMOL.
  buildCompleteDummyInventory (ioBomRoot);

  // ////// Fare Families Bom Tree ///////
  // Build the inventory (flight-dates) and the schedule (flight period)
  // parts with fare families.
  buildSampleInventoryScheduleForFareFamilies (ioBomRoot);

  // Build the pricing (fare rules) and revenue accounting (yields) parts.
  buildSamplePricingForFareFamilies (ioBomRoot);

  // Build a dummy inventory, needed by RMOL.
  buildCompleteDummyInventoryForFareFamilies (ioBomRoot);
}

// ////////////////////////////////////////////////////////////////////////
void CmdBomManager::buildSampleInventorySchedule (BomRoot& ioBomRoot) {

  // Inventory
  // Step 0.1: Inventory level
  // Create an Inventory for BA
  const AirlineCode_T lAirlineCodeBA ("BA");
  const InventoryKey lBAKey (lAirlineCodeBA);
  Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
  FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
  FacBomManager::linkWithParent (ioBomRoot, lBAInv);

  // Add the airline feature object to the BA inventory
  const AirlineFeatureKey lAirlineFeatureBAKey (lAirlineCodeBA);
  AirlineFeature& lAirlineFeatureBA =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureBAKey);
  FacBomManager::setAirlineFeature (lBAInv, lAirlineFeatureBA);
  FacBomManager::linkWithParent (lBAInv, lAirlineFeatureBA);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureBA);

  // Create an Inventory for AF
  const AirlineCode_T lAirlineCodeAF ("AF");
  const InventoryKey lAFKey (lAirlineCodeAF);
  Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
  FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
  FacBomManager::linkWithParent (ioBomRoot, lAFInv);

  // Add the airline feature object to the AF inventory
  const AirlineFeatureKey lAirlineFeatureAFKey (lAirlineCodeAF);
  AirlineFeature& lAirlineFeatureAF =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureAFKey);
  FacBomManager::setAirlineFeature (lAFInv, lAirlineFeatureAF);
  FacBomManager::linkWithParent (lAFInv, lAirlineFeatureAF);
  // Link the airline feature object with the top of the BOM tree
```

```
FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureAF);

// BA
// Step 0.2: Flight-date level
// Create a FlightDate (BA9/10-JUN-2011) for BA's Inventory
FlightNumber_T lFlightNumber = 9;
Date_T lDate (2011, 6, 10);
FlightDateKey lFlightDateKey (lFlightNumber, lDate);

FlightDate& lBA9_20110610_FD =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lBAInv, lBA9_20110610_FD);
FacBomManager::linkWithParent (lBAInv, lBA9_20110610_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());

// Step 0.3: Segment-date level
// Create a first SegmentDate (LHR-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
  arrier=BA&FlightNumber=0009&from=LHR&to=SYD&depDate=100611&SellingClass=O
const AirportCode_T lLHR ("LHR");
const AirportCode_T lSYD ("SYD");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l2135 (21, 45, 0);
const Duration_T l0610 (6, 10, 0);
const Duration_T l2205 (22, 05, 0);
SegmentDateKey lSegmentDateKey (lLHR, lSYD);

SegmentDate& lLHRSYDSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRSYDSegment);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRSYDSegment);

// Add the routing leg keys to the LHR-SYD segment.
const std::string lBALHRRoutingLegStr = "BA;9;2011-Jun-10;LHR";
const std::string lBABKKRoutingLegStr = "BA;9;2011-Jun-10;BKK";
lLHRSYDSegment.addLegKey (lBALHRRoutingLegStr);
lLHRSYDSegment.addLegKey (lBABKKRoutingLegStr);

// Fill the SegmentDate content
lLHRSYDSegment.setBoardingDate (lDate);
lLHRSYDSegment.setOffDate (lDate + l2Days);
lLHRSYDSegment.setBoardingTime (l2135);
lLHRSYDSegment.setOffTime (l0610);
lLHRSYDSegment.setElapsedTime (l2135);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRSYDSegment);

// Create a second SegmentDate (LHR-BKK) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
  arrier=BA&FlightNumber=0009&from=LHR&to=BKK&depDate=100611&SellingClass=O
const AirportCode_T lBKK ("BKK");
const Duration_T l1540 (15, 40, 0);
const Duration_T l1105 (11, 5, 0);
lSegmentDateKey = SegmentDateKey (lLHR, lBKK);

SegmentDate& lLHRBKKSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRBKKSegment);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRBKKSegment);

// Add the routing leg key to the LHR-BKK segment.
lLHRBKKSegment.addLegKey (lBALHRRoutingLegStr);
```

```
// Fill the SegmentDate content
lLHRBKKSegment.setBoardingDate (lDate);
lLHRBKKSegment.setOffDate (lDate + l1Day);
lLHRBKKSegment.setBoardingTime (l2135);
lLHRBKKSegment.setOffTime (l1540);
lLHRBKKSegment.setElapsedTime (l1105);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRBKKSegment);

// Create a third SegmentDate (BKK-SYD) for BA's Inventory
// See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&C
  arrier=BA&FlightNumber=0009&from=BKK&to=SYD&depDate=110611&SellingClass=O
const Duration_T l1705 (17, 5, 0);
const Duration_T l0905 (9, 5, 0);
lSegmentDateKey = SegmentDateKey (lBKK, lSYD);

SegmentDate& lBKKSYDSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKSYDSegment);
FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKSYDSegment);

// Add the routing leg key to the BKK-SYD segment.
lBKKSYDSegment.addLegKey (lBABKKRoutingLegStr);

// Fill the SegmentDate content
lBKKSYDSegment.setBoardingDate (lDate + l1Day);
lBKKSYDSegment.setOffDate (lDate + l2Days);
lBKKSYDSegment.setBoardingTime (l1705);
lBKKSYDSegment.setOffTime (l1540);
lBKKSYDSegment.setElapsedTime (l0905);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lBKKSYDSegment);

// Step 0.4: Leg-date level
// Create a first LegDate (LHR) for BA's Inventory
LegDateKey lLegDateKey (lLHR);

LegDate& lLHRLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRLeg);
FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRLeg);

// Fill the LegDate content
lLHRLeg.setOffPoint (lBKK);
lLHRLeg.setBoardingDate (lDate);
lLHRLeg.setOffDate (lDate + l1Day);
lLHRLeg.setBoardingTime (l2135);
lLHRLeg.setOffTime (l1540);
lLHRLeg.setElapsedTime (l1105);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lLHRLeg.toString());

// Create a second LegDate (BKK)
lLegDateKey = LegDateKey (lBKK);

LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKLeg);
FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKLeg);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lBKKLeg.toString());

// Fill the LegDate content
lBKKLeg.setOffPoint (lSYD);
lBKKLeg.setBoardingDate (lDate + l1Day);
```

```
lBKKLeg.setOffDate (lDate + l2Days);
lBKKLeg.setBoardingTime (l1705);
lBKKLeg.setOffTime (l1540);
lBKKLeg.setElapsedTime (l0905);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment LHR-BKK of BA's Inventory
const CabinCode_T lY ("Y");
SegmentCabinKey lYSegmentCabinKey (lY);

SegmentCabin& lLHRBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRBKKSegmentYCabin);
FacBomManager::linkWithParent (lLHRBKKSegment, lLHRBKKSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRBKKSegmentYCabin.toString());

// Create a SegmentCabin (Y) of the Segment BKK-SYD;
SegmentCabin& lBKKSYDSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lBKKSYDSegment, lBKKSYDSegmentYCabin);
FacBomManager::linkWithParent (lBKKSYDSegment, lBKKSYDSegmentYCabin);


// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lBKKSYDSegmentYCabin.toString());

// Create a SegmentCabin (Y) of the Segment LHR-SYD;
SegmentCabin& lLHRSYDSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRSYDSegmentYCabin);
FacBomManager::linkWithParent (lLHRSYDSegment, lLHRSYDSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRSYDSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
LegCabinKey lYLegCabinKey (lY);

LegCabin& lLHRLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lLHRLeg, lLHRLegYCabin);
FacBomManager::linkWithParent (lLHRLeg, lLHRLegYCabin);

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

// Create a LegCabin (Y) for the Leg BKK-SYD
LegCabin& lBKKLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lBKKLegYCabin.toString());

// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
const FamilyCode_T l1 ("EcoSaver");
FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lLHRBKKSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
                                lLHRBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lLHRBKKSegmentYCabin,
```

```
                                lLHRBKKSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lLHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1)  for the Segment BKK-SYD, cabin Y on BA's Inv
FareFamily& lBKKSYDSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                lBKKSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (lBKKSYDSegmentYCabin,
                                lBKKSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lLHRBKKSegmentYCabin1Family.toString());

// Create a FareFamily (1)  for the Segment LHR-SYD, cabin Y on BA's Inv
FareFamily& lLHRSYDSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1Family);
FacBomManager::linkWithParent (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lLHRBKKSegmentYCabin1Family.toString());


// Step 0.8: booking class level
// Create a BookingClass (Q) for the Segment LHR-BKK, cabin Y,
// fare family 1 on BA's Inv
const ClassCode_T lQ ("Q");
BookingClassKey lQBookingClassKey (lQ);

BookingClass& lLHRBKKSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin1Family,
                                lLHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lLHRBKKSegmentYCabin1Family,
                                lLHRBKKSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
                                lLHRBKKSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lLHRBKKSegment,
                                lLHRBKKSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lLHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lBKKSYDSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin1Family,
                                lBKKSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lBKKSYDSegmentYCabin1Family,
                                lBKKSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                lBKKSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lBKKSYDSegment,
                                lBKKSYDSegmentYCabin1FamilyQClass);
```

```
// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lLHRBKKSegmentYCabin1FamilyQClass.toString());

// Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
// fare family 1 on BA's Inv
BookingClass& lLHRSYDSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin1Family,
                                lLHRSYDSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lLHRSYDSegmentYCabin1Family,
                               lLHRSYDSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                lLHRSYDSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lLHRSYDSegment,
                                lLHRSYDSegmentYCabin1FamilyQClass);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lLHRBKKSegmentYCabin1FamilyQClass.toString());


// ////// AF ///////
// Step 0.2: Flight-date level
// Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
lFlightNumber = 84;
lDate = Date_T (2011, 3, 20);
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lAF084_20110320_FD =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lAFInv, lAF084_20110320_FD);
FacBomManager::linkWithParent (lAFInv, lAF084_20110320_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate (CDG-SFO) for AF's Inventory
const AirportCode_T lCDG ("CDG");
const AirportCode_T lSFO ("SFO");
const Duration_T l1040 (10, 40, 0);
const Duration_T l1250 (12, 50, 0);
const Duration_T l1110 (11, 10, 0);
lSegmentDateKey = SegmentDateKey (lCDG, lSFO);

SegmentDate& lCDGSFOSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGSFOSegment);
FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGSFOSegment);

// Add the routing leg key to the CDG-SFO segment.
const std::string lAFCDGRoutingLegStr = "AF;84;2011-Mar-20;CDG";
lCDGSFOSegment.addLegKey (lAFCDGRoutingLegStr);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lCDGSFOSegment.toString());

// Fill the SegmentDate content
lCDGSFOSegment.setBoardingDate (lDate);
lCDGSFOSegment.setOffDate (lDate);
lCDGSFOSegment.setBoardingTime (l1040);
lCDGSFOSegment.setOffTime (l1250);
lCDGSFOSegment.setElapsedTime (l1110);

// Step 0.4: Leg-date level
```

```
// Create a LegDate (CDG) for AF's Inventory
lLegDateKey = LegDateKey (lCDG);

LegDate& lCDGLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGLeg);
FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGLeg);

// Fill the LegDate content
lCDGLeg.setOffPoint (lSFO);
lCDGLeg.setBoardingDate (lDate);
lCDGLeg.setOffDate (lDate);
lCDGLeg.setBoardingTime (l1040);
lCDGLeg.setOffTime (l1250);
lCDGLeg.setElapsedTime (l1110);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment CDG-SFO of AF's Inventory
SegmentCabin& lCDGSFOSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGSFOSegmentYCabin);
FacBomManager::linkWithParent (lCDGSFOSegment, lCDGSFOSegmentYCabin);

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lCDGSFOSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg CDG-SFO on AF's Inventory
LegCabin& lCDGLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lCDGLeg, lCDGLegYCabin);
FacBomManager::linkWithParent (lCDGLeg, lCDGLegYCabin);

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

// Step 0.7: fare family level
// Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
FareFamily& lCDGSFOSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1Family);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
                               lCDGSFOSegmentYCabin1Family);

// Display the fare family
// STDAIR_LOG_DEBUG ("fareFamily: "
//
//                   << lCDGSFOSegmentYCabin1Family.toString());

// Step 0.8: booking class level Create a BookingClass (Q) for the
// Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
BookingClass& lCDGSFOSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin1Family,
                                lCDGSFOSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lCDGSFOSegmentYCabin1Family,
                               lCDGSFOSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                lCDGSFOSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lCDGSFOSegment,
                                lCDGSFOSegmentYCabin1FamilyQClass);

// Display the booking class
```

```
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lCDGSFOSegmentYCabin1FamilyQClass.toString());

/*==========================================================================
  =====
  ==========================================================================
  =====
  ==========================================================================
  =====*/
// Schedule:
// BA:
// Step 1: flight period level
// Create a flight period for BA9:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lBA9DateRangeStart (2010, boost::gregorian::Jun, 6);
const Date_T lBA9DateRangeEnd (2010, boost::gregorian::Jun, 7);
const DatePeriod_T lBA9DatePeriod (lBA9DateRangeStart, lBA9DateRangeEnd);
const PeriodStruct lBA9PeriodStruct (lBA9DatePeriod, lDoWSrtuct);

lFlightNumber = FlightNumber_T (9);

FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);

FlightPeriod& lBA9FlightPeriod =
  FacBom<FlightPeriod>::instance().create (lBA9FlightPeriodKey);
FacBomManager::addToListAndMap (lBAInv, lBA9FlightPeriod);
FacBomManager::linkWithParent (lBAInv, lBA9FlightPeriod);

// Step 2: segment period level
// Create a segment period for LHR-SYD:

SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);

SegmentPeriod& lLHRSYDSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lLHRSYDSegmentPeriodKey);
FacBomManager::addToListAndMap (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
FacBomManager::linkWithParent (lBA9FlightPeriod, lLHRSYDSegmentPeriod);

lLHRSYDSegmentPeriod.setBoardingTime (l2135);
lLHRSYDSegmentPeriod.setOffTime (l1540);
lLHRSYDSegmentPeriod.setElapsedTime (l1105);
ClassList_String_T lYM ("YM");
lLHRSYDSegmentPeriod.addCabinBookingClassList (lY,lYM);

// AF:
// Step 1: flight period level
// Create a flight period for AF84:
const Date_T lAF84DateRangeStart (2011, boost::gregorian::Mar, 20);
const Date_T lAF84DateRangeEnd (2011, boost::gregorian::Mar, 21);
const DatePeriod_T lAF84DatePeriod (lAF84DateRangeStart, lAF84DateRangeEnd);
const PeriodStruct lAF84PeriodStruct (lAF84DatePeriod, lDoWSrtuct);

lFlightNumber = FlightNumber_T (84);

FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);

FlightPeriod& lAF84FlightPeriod =
  FacBom<FlightPeriod>::instance().create (lAF84FlightPeriodKey);
FacBomManager::addToListAndMap (lAFInv, lAF84FlightPeriod);
FacBomManager::linkWithParent (lAFInv, lAF84FlightPeriod);

// Step 2: segment period level
// Create a segment period for CDG-SFO:

SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);

SegmentPeriod& lCDGSFOSegmentPeriod =
```

```
    FacBom<SegmentPeriod>::instance().create (lCDGSFOSegmentPeriodKey);
  FacBomManager::addToListAndMap (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
  FacBomManager::linkWithParent (lAF84FlightPeriod, lCDGSFOSegmentPeriod);

  lCDGSFOSegmentPeriod.setBoardingTime (l1040);
  lCDGSFOSegmentPeriod.setOffTime (l1250);
  lCDGSFOSegmentPeriod.setElapsedTime (l1110);
  lCDGSFOSegmentPeriod.addCabinBookingClassList (lY,lYM);

  /*=========================================================================
    =====
    =========================================================================
    =====
    =========================================================================
    =====*/
  // O&D
  // Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory
  OnDString_T  lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
  OnDStringList_T lBAOnDStrList;
  lBAOnDStrList.push_back (lBALHRSYDOnDStr);

  OnDDateKey lBAOnDDateKey (lBAOnDStrList);
  OnDDate& lBA_LHRSYD_OnDDate =
    FacBom<OnDDate>::instance().create (lBAOnDDateKey);
  // Link to the inventory
  FacBomManager::addToListAndMap (lBAInv, lBA_LHRSYD_OnDDate);
  FacBomManager::linkWithParent (lBAInv, lBA_LHRSYD_OnDDate);

  // Add the segment
  FacBomManager::addToListAndMap (lBA_LHRSYD_OnDDate, lLHRSYDSegment);

  // Add total forecast info for cabin Y.
  const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
  const WTP_T lWTP750 = 750.0;
  const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
  lBA_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

  // Create an O&D Date (AF;84,2011-Mar-21;CDG,SFO) for AF's Inventory
  OnDString_T lAFLHRSYDOnDStr = "AF;9,2011-Mar-20;CDG,SFO";
  OnDStringList_T lAFOnDStrList;
  lAFOnDStrList.push_back (lAFLHRSYDOnDStr);

  OnDDateKey lAFOnDDateKey (lAFOnDStrList);
  OnDDate& lAF_LHRSYD_OnDDate =
    FacBom<OnDDate>::instance().create (lAFOnDDateKey);
  // Link to the inventory
  FacBomManager::addToListAndMap (lAFInv, lAF_LHRSYD_OnDDate);
  FacBomManager::linkWithParent (lAFInv, lAF_LHRSYD_OnDDate);

  // Add the segment
  FacBomManager::addToListAndMap (lAF_LHRSYD_OnDDate, lLHRSYDSegment);

  // Add total forecast info for cabin Y.
  lAF_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

}

// ////////////////////////////////////////////////////////////////////////
void CmdBomManager::
buildSampleInventoryScheduleForFareFamilies (BomRoot& ioBomRoot) {

  // Inventory
  // Step 0.1: Inventory level
  // Get the Inventory SQ (already built by construction)
  const InventoryKey lSQKey ("SQ");
  Inventory& lSQInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                       lSQKey.toString());
```

```
// SQ
// Step 0.2: Flight-date level
// Create a FlightDate (SQ747/8-FEB-2010) for SQ's Inventory
const FlightNumber_T lFlightNumber747 = 747;
const Date_T lDate (2010, 2, 8);
const FlightDateKey lFlightDateKey (lFlightNumber747, lDate);

FlightDate& lSQ747_20100208_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ747_20100208_FD);
FacBomManager::linkWithParent (lSQInv, lSQ747_20100208_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lSQ747_20100208_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate (SIN-BKK) for SQ's Inventory
const AirportCode_T lSIN ("SIN");
const AirportCode_T lBKK ("BKK");
const Duration_T l0635 (6, 35, 0);
const Duration_T l0800 (8, 0, 0);
const Duration_T l0225 (2, 25, 0);
const SegmentDateKey lSegmentDateKey (lSIN, lBKK);

SegmentDate& lSINBKKSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINBKKSegment);
FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINBKKSegment);

// Add the routing leg key to the SIN-BKK segment.
const std::string lSQSINRoutingLegStr = "SQ;747;2010-Feb-8;SIN";
lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

// Fill the SegmentDate content
lSINBKKSegment.setBoardingDate (lDate);
lSINBKKSegment.setOffDate (lDate);
lSINBKKSegment.setBoardingTime (l0635);
lSINBKKSegment.setOffTime (l0800);
lSINBKKSegment.setElapsedTime (l0225);

// Display the segment-date
// STDAIR_LOG_DEBUG ("SegmentDate: " << lSINBKKSegment);

// Step 0.4: Leg-date level
// Create a LegDate (SIN) for SQ's Inventory
const LegDateKey lLegDateKey (lSIN);

LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINLeg);
FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINLeg);

// Fill the LegDate content
lSINLeg.setOffPoint (lBKK);
lSINLeg.setBoardingDate (lDate);
lSINLeg.setOffDate (lDate);
lSINLeg.setBoardingTime (l0635);
lSINLeg.setOffTime (l0800);
lSINLeg.setElapsedTime (l0225);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lSINLeg.toString());

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
const CabinCode_T lY ("Y");
const SegmentCabinKey lYSegmentCabinKey (lY);
```

```
SegmentCabin& lSINBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
lSINBKKSegmentYCabin.activateFareFamily ();

// Display the segment-cabin
// STDAIR_LOG_DEBUG ("SegmentCabin: " << lSINBKKSegmentYCabin.toString());

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
const LegCabinKey lYLegCabinKey (lY);
LegCabin& lSINLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);

// Display the leg-cabin
// STDAIR_LOG_DEBUG ("LegCabin: " << lSINLegYCabin.toString());

// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l1 ("1");
const FareFamilyKey l1FareFamilyKey (l1);
FareFamily& lSINBKKSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                               lSINBKKSegmentYCabin1Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lSINBKKSegmentYCabin1Family.toString());

// Create a FareFamily (2) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l2 ("2");
const FareFamilyKey l2FareFamilyKey (l2);
FareFamily& lSINBKKSegmentYCabin2Family =
    FacBom<FareFamily>::instance().create (l2FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin2Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                               lSINBKKSegmentYCabin2Family);

// Display the booking class
// STDAIR_LOG_DEBUG ("FareFamily: "
//                   << lSINBKKSegmentYCabin2Family.toString());

// Step 0.8: booking class level
// Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
// fare family 2 on SQ's Inv
const ClassCode_T lClassY ("Y");
const BookingClassKey lYBookingClassKey (lClassY);
BookingClass& lSINBKKSegmentYCabin2FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                lSINBKKSegmentYCabin2FamilyYClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
                               lSINBKKSegmentYCabin2FamilyYClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin2FamilyYClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin2FamilyYClass);
lSINBKKSegmentYCabin2FamilyYClass.setYield(1200);
```

```
// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lSINBKKSegmentYCabin2FamilyYClass.toString());

// Create a BookingClass (B) for the Segment SIN-BKK, cabin Y,
// fare family 2 on SQ's Inv
const ClassCode_T lB ("B");
const BookingClassKey lBBookingClassKey (lB);
BookingClass& lSINBKKSegmentYCabin2FamilyBClass =
  FacBom<BookingClass>::instance().create (lBBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                lSINBKKSegmentYCabin2FamilyBClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
                               lSINBKKSegmentYCabin2FamilyBClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin2FamilyBClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin2FamilyBClass);
lSINBKKSegmentYCabin2FamilyBClass.setYield(800);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lSINBKKSegmentYCabin2FamilyBClass.toString());

// Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
const ClassCode_T lM ("M");
const BookingClassKey lMBookingClassKey (lM);
BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                               lSINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyMClass);
lSINBKKSegmentYCabin1FamilyMClass.setYield(900);

// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lSINBKKSegmentYCabin1FamilyMClass.toString());

// Create a BookingClass (Q) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
const ClassCode_T lQ ("Q");
const BookingClassKey lQBookingClassKey (lQ);
BookingClass& lSINBKKSegmentYCabin1FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyQClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                               lSINBKKSegmentYCabin1FamilyQClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyQClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyQClass);
lSINBKKSegmentYCabin1FamilyQClass.setYield(600);


// Display the booking class
// STDAIR_LOG_DEBUG ("BookingClass: "
//                   << lSINBKKSegmentYCabin1FamilyQClass.toString());
```

```
  /*==========================================================================
    =====
    ==========================================================================
    =====
    ==========================================================================
    =====*/
  // Schedule:
  // SQ:
  // Step 1: flight period level
  // Create a flight period for SQ747:
  const DoWStruct lDoWSrtuct ("1111111");
  const Date_T lSQ747DateRangeStart (2010, boost::gregorian::Feb, 8);
  const Date_T lSQ747DateRangeEnd (2010, boost::gregorian::Feb, 9);
  const DatePeriod_T lSQ747DatePeriod (lSQ747DateRangeStart,
                                       lSQ747DateRangeEnd);
  const PeriodStruct lSQ747PeriodStruct (lSQ747DatePeriod, lDoWSrtuct);

  const FlightPeriodKey lSQ747FlightPeriodKey (lFlightNumber747,
                                               lSQ747PeriodStruct);
  FlightPeriod& lSQ747FlightPeriod =
    FacBom<FlightPeriod>::instance().create (lSQ747FlightPeriodKey);
  FacBomManager::addToListAndMap (lSQInv, lSQ747FlightPeriod);
  FacBomManager::linkWithParent (lSQInv, lSQ747FlightPeriod);

  // Step 2: segment period level
  // Create a segment period for SIN-BKK:

  const SegmentPeriodKey lSINBKKSegmentPeriodKey (lSIN, lBKK);
  SegmentPeriod& lSINBKKSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lSINBKKSegmentPeriodKey);
  FacBomManager::addToListAndMap (lSQ747FlightPeriod, lSINBKKSegmentPeriod);
  FacBomManager::linkWithParent (lSQ747FlightPeriod, lSINBKKSegmentPeriod);

  ClassList_String_T lYBMQ  ("YBMQ");
  lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYBMQ);
  lSINBKKSegmentPeriod.setBoardingTime (l0635);
  lSINBKKSegmentPeriod.setOffTime (l0800);
  lSINBKKSegmentPeriod.setElapsedTime (l0225);

  /*==========================================================================
    =====
    ==========================================================================
    =====
    ==========================================================================
    =====*/
  // O&D
  // Create an O&D Date (SQ;747,2011-Feb-14;SIN,BKK) for SQ's Inventory
  const OnDString_T lSQSINBKKOnDStr = "SQ;747,2011-Feb-14;SIN,BKK";
  OnDStringList_T lSQOnDStrList;
  lSQOnDStrList.push_back (lSQSINBKKOnDStr);

  const OnDDateKey lSQOnDDateKey (lSQOnDStrList);
  OnDDate& lSQ_SINBKK_OnDDate =
    FacBom<OnDDate>::instance().create (lSQOnDDateKey);
  // Link to the inventory
  FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
  FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);
  // Add total forecast info for cabin Y.
  const MeanStdDevPair_T lMean120StdDev12 (120.0, 12.0);
  const WTP_T lWTP1000 = 1000.0;
  const WTPDemandPair_T lWTP1000Mean120StdDev12 (lWTP1000, lMean120StdDev12);
  lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP1000Mean120StdDev12);

  // Add the segment
  FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
}
```

```
// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildDummyLegSegmentAccesses (BomRoot& ioBomRoot) {

  /* Build the direct accesses between the dummy segment cabins and the dummy
leg cabins within the dummy flight dates (the dummy fare family
flight date and the classic dummy flight date).

As for now (May 2012), that method is called only by RMOL.
It is a substitute for the code doing it automatically located in AirInv.
See the AIRINV::InventoryManager::createDirectAccesses command.
   */

  // ////// Dummy Inventory Leg Segment Accesses ///////
  // Retrieve the (sample) segment-cabin.
  SegmentCabin& lDummySegmentCabin =
    BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

  // Retrieve the (sample) leg-cabin.
  LegCabin& lDummyLegCabin =
    BomRetriever::retrieveDummyLegCabin (ioBomRoot);

  // Links between the segment-date and the leg-date
  FacBomManager::addToListAndMap (lDummyLegCabin, lDummySegmentCabin);
  FacBomManager::addToListAndMap (lDummySegmentCabin, lDummyLegCabin);

  // ////// Fare Families Dummy Inventory Leg Segment Accesses ///////
  const bool isForFareFamilies = true;
  // Retrieve the (sample) segment-cabin for fare families.
  SegmentCabin& lFFDummySegmentCabin =
    BomRetriever::retrieveDummySegmentCabin (ioBomRoot, isForFareFamilies);

  // Retrieve the (sample) leg-cabin for fare families.
  stdair::LegCabin& lFFDummyLegCabin =
    stdair::BomRetriever::retrieveDummyLegCabin (ioBomRoot,
                                                 isForFareFamilies);

  // Links between the segment-date and the leg-date for fare families.
  FacBomManager::addToListAndMap (lFFDummyLegCabin, lFFDummySegmentCabin);
  FacBomManager::addToListAndMap (lFFDummySegmentCabin, lFFDummyLegCabin);
}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildCompleteDummyInventory  (BomRoot& ioBomRoot) {

  // Build a dummy inventory, containing a dummy flight-date with a
  // single segment-cabin and a single leg-cabin.
  const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
  buildDummyInventory (ioBomRoot, lCapacity);

  // Retrieve the (sample) segment-cabin.
  SegmentCabin& lDummySegmentCabin =
    BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

  // Retrieve the (sample) leg-cabin.
  LegCabin& lDummyLegCabin =
    BomRetriever::retrieveDummyLegCabin (ioBomRoot);

  // Add some booking classes to the dummy segment-cabin and some
  // virtual ones to the dummy leg-cabin.
  // First booking class yield and demand information.
  Yield_T lYield = 100;
  MeanValue_T lMean = 20;
  StdDevValue_T lStdDev= 9;
  BookingClassKey lBCKey (DEFAULT_CLASS_CODE);

  BookingClass& lDummyBookingClass =
```

```
    FacBom<BookingClass>::instance().create (lBCKey);
  lDummyBookingClass.setYield (lYield);
  lDummyBookingClass.setMean (lMean);
  lDummyBookingClass.setStdDev (lStdDev);
  // Add a booking class to the segment-cabin.
  FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
  BookingClassList_T lDummyBookingClassList;
  lDummyBookingClassList.push_back(&lDummyBookingClass);

  VirtualClassStruct lDummyVirtualClass (lDummyBookingClassList);
  lDummyVirtualClass.setYield (lYield);
  lDummyVirtualClass.setMean (lMean);
  lDummyVirtualClass.setStdDev (lStdDev);
  // Add the corresponding virtual class to the leg-cabin.
  lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

  // Second booking class yield and demand information.
  lYield = 70;
  lMean = 45;
  lStdDev= 12;
  lDummyBookingClass.setYield (lYield);
  lDummyBookingClass.setMean (lMean);
  lDummyBookingClass.setStdDev (lStdDev);
  // Add a booking class to the segment-cabin.
  FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

  lDummyVirtualClass.setYield (lYield);
  lDummyVirtualClass.setMean (lMean);
  lDummyVirtualClass.setStdDev (lStdDev);
  // Add the corresponding virtual class to the leg-cabin.
  lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

  // Third booking class yield and demand information.
  lYield = 42;
  lMean = 80;
  lStdDev= 16;
  lDummyBookingClass.setYield (lYield);
  lDummyBookingClass.setMean (lMean);
  lDummyBookingClass.setStdDev (lStdDev);
  // Add a booking class to the segment-cabin.
  FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

  lDummyVirtualClass.setYield (lYield);
  lDummyVirtualClass.setMean (lMean);
  lDummyVirtualClass.setStdDev (lStdDev);
  // Add the corresponding virtual class to the leg-cabin.
  lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildDummyInventory (BomRoot& ioBomRoot,
                                         const CabinCapacity_T& iCapacity) {
  // Inventory
  const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
  Inventory& lInv = FacBom<Inventory>::instance().create (lInventoryKey);
  FacBomManager::addToListAndMap (ioBomRoot, lInv);
  FacBomManager::linkWithParent (ioBomRoot, lInv);

  // Add the airline feature object to the dummy inventory
  const AirlineFeatureKey lAirlineFeatureKey (DEFAULT_AIRLINE_CODE);
  AirlineFeature& lAirlineFeature =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureKey);
  FacBomManager::setAirlineFeature (lInv, lAirlineFeature);
  FacBomManager::linkWithParent (lInv, lAirlineFeature);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);
```

```
// Flight-date
FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE);
FlightDate& lFlightDate =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lInv, lFlightDate);
FacBomManager::linkWithParent (lInv, lFlightDate);

// Leg-date
LegDateKey lLegDateKey (DEFAULT_ORIGIN);
LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lFlightDate, lLeg);
FacBomManager::linkWithParent (lFlightDate, lLeg);

// Fill the LegDate content
lLeg.setOffPoint (DEFAULT_DESTINATION);
lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
lLeg.setBoardingTime (Duration_T (14, 0, 0));
lLeg.setOffTime (Duration_T (16, 0, 0));
lLeg.setElapsedTime (Duration_T (8, 0, 0));

// Leg-cabin
LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
FacBomManager::addToListAndMap (lLeg, lLegCabin);
FacBomManager::linkWithParent (lLeg, lLegCabin);

lLegCabin.setCapacities (iCapacity);
lLegCabin.setAvailabilityPool (iCapacity);

// Segment-date
SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
SegmentDate& lSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lFlightDate, lSegment);
FacBomManager::linkWithParent (lFlightDate, lSegment);

// Add the routing leg key to the dummy segment.
std::ostringstream oStr;
oStr << DEFAULT_AIRLINE_CODE << ";"
     << DEFAULT_FLIGHT_NUMBER << ";"
     << DEFAULT_DEPARTURE_DATE << ";"
     << DEFAULT_ORIGIN;
lSegment.addLegKey (oStr.str());

// Fill the SegmentDate content
lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
lSegment.setBoardingTime (Duration_T (14, 0, 0));
lSegment.setOffTime (Duration_T (16, 0, 0));
lSegment.setElapsedTime (Duration_T (8, 0, 0));

// Segment-cabin
SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
SegmentCabin& lSegmentCabin =
  FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
FacBomManager::linkWithParent (lSegment, lSegmentCabin);

// Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
const FamilyCode_T l1 ("EcoSaver");
FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
```

```
  FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);

  // Create a booking-class
  const ClassCode_T lQ ("Q");
  BookingClassKey lQBookingClassKey (lQ);

  BookingClass& lSegmentYCabin1FamilyQClass =
    FacBom<BookingClass>::instance().create (lQBookingClassKey);
  FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                  lSegmentYCabin1FamilyQClass);
  FacBomManager::linkWithParent (lSegmentYCabin1Family,
                                 lSegmentYCabin1FamilyQClass);

  FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyQClass);
  FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyQClass);

  /*==========================================================================
    =====
    ==========================================================================
    =====
    ==========================================================================
    =====*/
  // Schedule:
  // XX:
  // Step 1: flight period level
  // Create a flight period for XX:
  const DoWStruct lDoWSrtuct ("1111111");
  const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
  const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
  const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
  const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);

  FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER, lXXPeriodStruct);

  FlightPeriod& lXXFlightPeriod =
    FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
  FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
  FacBomManager::linkWithParent (lInv, lXXFlightPeriod);

  // Step 2: segment period level
  // Create a segment period

  SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);

  SegmentPeriod& lXXSegmentPeriod =
    FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
  FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
  FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);

  lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
  lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
  lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
  const CabinCode_T lY ("Y");
  const ClassList_String_T lYQ ("YQ");
  lXXSegmentPeriod.addCabinBookingClassList (lY,lYQ);


}

// ////////////////////////////////////////////////////////////////////////
void CmdBomManager::
buildCompleteDummyInventoryForFareFamilies (BomRoot& ioBomRoot) {

  // Build a dummy inventory, containing a dummy flight-date with a
  // single segment-cabin and a single leg-cabin (for fare families
  // algorithms)
```

```
// Get the default Inventory object (already built in by construction)
const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
Inventory& lInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                   lInventoryKey.toString());

// Create a dummy Flight-date
const FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER_FF,
                                   DEFAULT_DEPARTURE_DATE);
FlightDate& lFlightDate =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lInv, lFlightDate);
FacBomManager::linkWithParent (lInv, lFlightDate);

// Create a dummy Leg-date
LegDateKey lLegDateKey (DEFAULT_ORIGIN);
LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lFlightDate, lLeg);
FacBomManager::linkWithParent (lFlightDate, lLeg);

// Fill the LegDate content
lLeg.setOffPoint (DEFAULT_DESTINATION);
lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
lLeg.setBoardingTime (Duration_T (14, 0, 0));
lLeg.setOffTime (Duration_T (16, 0, 0));
lLeg.setElapsedTime (Duration_T (8, 0, 0));

// Create a dummy Leg-cabin
const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
FacBomManager::addToListAndMap (lLeg, lLegCabin);
FacBomManager::linkWithParent (lLeg, lLegCabin);
const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
lLegCabin.setCapacities (lCapacity);
lLegCabin.setAvailabilityPool (lCapacity);

// Create a dummy Segment-date
const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
SegmentDate& lSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lFlightDate, lSegment);
FacBomManager::linkWithParent (lFlightDate, lSegment);

// Add the routing leg key to the dummy segment.
std::ostringstream oStr;
oStr << DEFAULT_AIRLINE_CODE << ";"
     << DEFAULT_FLIGHT_NUMBER << ";"
     << DEFAULT_DEPARTURE_DATE << ";"
     << DEFAULT_ORIGIN;
lSegment.addLegKey (oStr.str());

// Fill the SegmentDate content
lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
lSegment.setBoardingTime (Duration_T (14, 0, 0));
lSegment.setOffTime (Duration_T (16, 0, 0));
lSegment.setElapsedTime (Duration_T (8, 0, 0));

// Create a dummy Segment-cabin
const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
SegmentCabin& lSegmentCabin =
  FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
FacBomManager::linkWithParent (lSegment, lSegmentCabin);

// Create a dummy FareFamily (FF1)
const FamilyCode_T l1 ("FF1");
```

```
const FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
// Set the forecasted demand
// TODO change the size (hard code)
MeanStdDevPairVector_T lDemandVector1FareFamily;
const unsigned int size = 16;
for (unsigned int idx = 0; idx < size; ++idx) {
  double i = static_cast<double> (idx);
  MeanStdDevPair_T lMeanStdDevPair (i/4.0, i/20.0);
  lDemandVector1FareFamily.push_back(lMeanStdDevPair);
}
lSegmentYCabin1Family.setMeanStdDev(lDemandVector1FareFamily);
FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);

// Create a dummy booking-class
const ClassCode_T lY ("Y");
const BookingClassKey lYBookingClassKey (lY);

BookingClass& lSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
Yield_T lYield = 1000;
lSegmentYCabin1FamilyYClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                lSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lSegmentYCabin1Family,
                               lSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyYClass);

// Create a second dummy booking-class
const ClassCode_T lU ("U");
const BookingClassKey lUBookingClassKey (lU);

BookingClass& lSegmentYCabin1FamilyUClass =
  FacBom<BookingClass>::instance().create (lUBookingClassKey);
lYield = 600;
lSegmentYCabin1FamilyUClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                lSegmentYCabin1FamilyUClass);
FacBomManager::linkWithParent (lSegmentYCabin1Family,
                               lSegmentYCabin1FamilyUClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyUClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyUClass);

// Create a second dummy FareFamily (2)
const FamilyCode_T l2 ("FF2");
const FareFamilyKey l2FareFamilyKey (l2);

FareFamily& lSegmentYCabin2Family =
  FacBom<FareFamily>::instance().create (l2FareFamilyKey);
// Set the forecasted demand
// TODO change the size (hard code)
MeanStdDevPairVector_T lDemandVector2FareFamily;
for (unsigned int idx = 0; idx < size; ++idx) {
  double i = static_cast<double> (idx);
  MeanStdDevPair_T lMeanStdDevPair (i/2.0, i/10.0);
  lDemandVector2FareFamily.push_back(lMeanStdDevPair);
}
lSegmentYCabin2Family.setMeanStdDev(lDemandVector2FareFamily);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2Family);
FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin2Family);
```

```
// Create a third dummy booking-class
const ClassCode_T lO ("O");
const BookingClassKey lOBookingClassKey (lO);

BookingClass& lSegmentYCabin2FamilyOClass =
  FacBom<BookingClass>::instance().create (lOBookingClassKey);
lYield = 750;
lSegmentYCabin2FamilyOClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin2Family,
                                lSegmentYCabin2FamilyOClass);
FacBomManager::linkWithParent (lSegmentYCabin2Family,
                               lSegmentYCabin2FamilyOClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyOClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyOClass);

 // Create a fourth dummy booking-class
const ClassCode_T lQ ("Q");
const BookingClassKey lQBookingClassKey (lQ);

BookingClass& lSegmentYCabin2FamilyQClass =
  FacBom<BookingClass>::instance().create (lQBookingClassKey);
lYield = 400;
lSegmentYCabin2FamilyQClass.setYield(lYield);
FacBomManager::addToListAndMap (lSegmentYCabin2Family,
                                lSegmentYCabin2FamilyQClass);
FacBomManager::linkWithParent (lSegmentYCabin2Family,
                               lSegmentYCabin2FamilyQClass);

FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyQClass);
FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyQClass);


/*==========================================================================
  =====
  ==========================================================================
  =====
  ==========================================================================
  =====*/
// Schedule:
// XX:
// Step 1: flight period level
// Create a flight period for XX:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);

const FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER_FF,
                                          lXXPeriodStruct);

FlightPeriod& lXXFlightPeriod =
  FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
FacBomManager::linkWithParent (lInv, lXXFlightPeriod);

// Step 2: segment period level
// Create a segment period
const SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN,
                                            DEFAULT_DESTINATION);

SegmentPeriod& lXXSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);
```

```
  lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
  lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
  lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
  const CabinCode_T lYCabin ("Y");
  const ClassList_String_T lYUOQ ("YUOQ");
  lXXSegmentPeriod.addCabinBookingClassList (lYCabin,lYUOQ);

}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildSamplePricing (BomRoot& ioBomRoot) {

  // Set the airport-pair primary key.
  const AirportPairKey lAirportPairKey (AIRPORT_LHR, AIRPORT_SYD);

  // Create the AirportPairKey object and link it to the BOM tree root.
  AirportPair& lAirportPair =
    FacBom<AirportPair>::instance().create (lAirportPairKey);
  FacBomManager::addToListAndMap (ioBomRoot, lAirportPair);
  FacBomManager::linkWithParent (ioBomRoot, lAirportPair);

  // Set the fare date-period primary key.
  const Date_T lDateRangeStart (2011, boost::gregorian::Jan, 15);
  const Date_T lDateRangeEnd (2011, boost::gregorian::Dec, 31);
  const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
  const DatePeriodKey lDatePeriodKey (lDateRange);

  // Create the DatePeriodKey object and link it to the PosChannel object.
  DatePeriod& lDatePeriod =
    FacBom<DatePeriod>::instance().create (lDatePeriodKey);
  FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
  FacBomManager::linkWithParent (lAirportPair, lDatePeriod);

  // Set the point-of-sale-channel primary key.
  const PosChannelKey lPosChannelKey (POS_LHR, CHANNEL_DN);

  // Create the PositionKey object and link it to the AirportPair object.
  PosChannel& lPosChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
  FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
  FacBomManager::linkWithParent (lDatePeriod, lPosChannel);

  // Set the fare time-period primary key.
  const Time_T lTimeRangeStart (0, 0, 0);
  const Time_T lTimeRangeEnd (23, 0, 0);
  const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);

  // Create the TimePeriodKey and link it to the DatePeriod object.
  TimePeriod& lTimePeriod =
    FacBom<TimePeriod>::instance().create (lTimePeriodKey);
  FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
  FacBomManager::linkWithParent (lPosChannel, lTimePeriod);

  // Pricing -- Generate the FareRule
  const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                          NO_ADVANCE_PURCHASE,
                                          SATURDAY_STAY,
                                          CHANGE_FEES,
                                          NON_REFUNDABLE,
                                          NO_STAY_DURATION);

  // Create the FareFeaturesKey and link it to the TimePeriod object.
  FareFeatures& lFareFeatures =
    FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
  FacBomManager::addToListAndMap (lTimePeriod, lFareFeatures);
  FacBomManager::linkWithParent (lTimePeriod, lFareFeatures);
```

```
  // Revenue Accounting -- Generate the YieldRule
  const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                            CABIN_Y);

  // Create the YieldFeaturesKey and link it to the TimePeriod object.
  YieldFeatures& lYieldFeatures =
    FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
  FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
  FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);

  // Generate Segment Features and link them to their respective
  // fare and yield rules.
  AirlineCodeList_T lAirlineCodeList;
  lAirlineCodeList.push_back (AIRLINE_CODE_BA);
  ClassList_StringList_T lClassCodeList;
  lClassCodeList.push_back (CLASS_CODE_Y);
  const AirlineClassListKey lAirlineClassListKey (lAirlineCodeList,
                                                  lClassCodeList);

  // Create the AirlineClassList
  AirlineClassList& lAirlineClassList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassListKey);
  // Link the AirlineClassList to the FareFeatures object
  lAirlineClassList.setFare (900);
  FacBomManager::addToListAndMap (lFareFeatures, lAirlineClassList);
  FacBomManager::linkWithParent (lFareFeatures, lAirlineClassList);

  // Link the AirlineClassList to the YieldFeatures object
  lAirlineClassList.setYield (900);
  FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassList);
  // \todo (gsabatier): the following calls overrides the parent for
  //       lAirlineClassList. Check that it is what is actually wanted.
  FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassList);
}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildSamplePricingForFareFamilies (BomRoot& ioBomRoot) {

  // Get the airport-pair primary key SIN-BKK
  // (already built by construction)
  const AirportPairKey lAirportPairKey ("SIN", "BKK");
  AirportPair& lAirportPair =
    BomManager::getObject<AirportPair>(ioBomRoot, lAirportPairKey.toString());

  // Set the fare date-period primary key.
  const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 1);
  const Date_T lDateRangeEnd (2011, boost::gregorian::Feb, 15);
  const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
  const DatePeriodKey lDatePeriodKey (lDateRange);

  // Create the DatePeriodKey object and link it to the PosChannel object.
  DatePeriod& lDatePeriod =
    FacBom<DatePeriod>::instance().create (lDatePeriodKey);
  FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
  FacBomManager::linkWithParent (lAirportPair, lDatePeriod);

  // Set the point-of-sale-channel primary key.
  const PosChannelKey lPosChannelKey ("SIN", CHANNEL_IN);

  // Create the PositionKey object and link it to the AirportPair object.
  PosChannel& lPosChannel =
    FacBom<PosChannel>::instance().create (lPosChannelKey);
  FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
  FacBomManager::linkWithParent (lDatePeriod, lPosChannel);

  // Set the fare time-period primary key.
```

```
const Time_T lTimeRangeStart (0, 0, 0);
const Time_T lTimeRangeEnd (23, 0, 0);
const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lTimePeriod =
  FacBom<TimePeriod>::instance().create (lTimePeriodKey);
FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
FacBomManager::linkWithParent (lPosChannel, lTimePeriod);

// Pricing -- Generate the FareRule
const DayDuration_T ONE_MONTH_ADVANCE_PURCHASE = 30;
// Generate the first FareFeatures for the class Q
const FareFeaturesKey lFareFeaturesQKey (TRIP_TYPE_ONE_WAY,
                                         ONE_MONTH_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lFareFeaturesQ =
  FacBom<FareFeatures>::instance().create (lFareFeaturesQKey);
FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesQ);
FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesQ);

// Generate the second FareFeatures for the class M
const FareFeaturesKey lFareFeaturesMKey (TRIP_TYPE_ONE_WAY,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         CHANGE_FEES,
                                         NON_REFUNDABLE,
                                         NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lFareFeaturesM =
  FacBom<FareFeatures>::instance().create (lFareFeaturesMKey);
FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesM);
FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesM);

// Generate the third FareFeatures for the class B
const FareFeaturesKey lFareFeaturesBKey (TRIP_TYPE_ONE_WAY,
                                          ONE_MONTH_ADVANCE_PURCHASE,
                                          SATURDAY_STAY,
                                          NO_CHANGE_FEES,
                                          NO_NON_REFUNDABLE,//Refundable
                                          NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lFareFeaturesB =
  FacBom<FareFeatures>::instance().create (lFareFeaturesBKey);
FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesB);
FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesB);

// Generate the fourth FareFeatures for the class Y
const FareFeaturesKey lFareFeaturesYKey (TRIP_TYPE_ONE_WAY,
                                         NO_ADVANCE_PURCHASE,
                                         SATURDAY_STAY,
                                         NO_CHANGE_FEES,
                                         NO_NON_REFUNDABLE,//Refundable
                                         NO_STAY_DURATION);

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lFareFeaturesY =
  FacBom<FareFeatures>::instance().create (lFareFeaturesYKey);
FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesY);
FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesY);
```

```
// Revenue Accounting -- Generate the YieldRule
const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
                                          CABIN_Y);

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lYieldFeatures =
  FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);

// Generate Segment Features and link them to their respective
// fare and yield rules.
AirlineCodeList_T lAirlineCodeList;
lAirlineCodeList.push_back ("SQ");

ClassList_StringList_T lClassYList;
lClassYList.push_back (CLASS_CODE_Y);
const AirlineClassListKey lAirlineClassYListKey (lAirlineCodeList,
                                                 lClassYList);

// Create the AirlineClassList
AirlineClassList& lAirlineClassYList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassYListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesY, lAirlineClassYList);
FacBomManager::linkWithParent (lFareFeaturesY, lAirlineClassYList);
lAirlineClassYList.setFare (1200);
lAirlineClassYList.setYield (1200);

// Link the AirlineClassList to the YieldFeatures object
FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassYList);
// \todo (gsabatier): the following calls overrides the parent for
//       lAirlineClassList. Check that it is what is actually wanted.
FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassYList);

ClassList_StringList_T lClassBList;
lClassBList.push_back ("B");
const AirlineClassListKey lAirlineClassBListKey (lAirlineCodeList,
                                                 lClassBList);
// Create the AirlineClassList
AirlineClassList& lAirlineClassBList =
  FacBom<AirlineClassList>::instance().create (lAirlineClassBListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesB, lAirlineClassBList);
FacBomManager::linkWithParent (lFareFeaturesB, lAirlineClassBList);
lAirlineClassBList.setFare (800);
lAirlineClassBList.setYield (800);

// Link the AirlineClassList to the YieldFeatures object
FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassBList);
// \todo (gsabatier): the following calls overrides the parent for
//       lAirlineClassList. Check that it is what is actually wanted.
FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassBList);

ClassList_StringList_T lClassMList;
lClassMList.push_back ("M");
const AirlineClassListKey lAirlineClassMListKey (lAirlineCodeList,
                                                 lClassMList);

// Create the AirlineClassList
AirlineClassList& lAirlineClassMList =
  FacBom<AirlineClassList>::instance().create (lAirlineClassMListKey);
// Link the AirlineClassList to the FareFeatures object
FacBomManager::addToListAndMap (lFareFeaturesM, lAirlineClassMList);
FacBomManager::linkWithParent (lFareFeaturesM, lAirlineClassMList);
lAirlineClassMList.setFare (900);
```

```
  lAirlineClassMList.setYield (900);

  // Link the AirlineClassList to the YieldFeatures object
  FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassMList);
  // \todo (gsabatier): the following calls overrides the parent for
  //       lAirlineClassList. Check that it is what is actually wanted.
  FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassMList);

  ClassList_StringList_T lClassQList;
  lClassQList.push_back ("Q");
  const AirlineClassListKey lAirlineClassQListKey (lAirlineCodeList,
                                                   lClassQList);

  // Create the AirlineClassList
  AirlineClassList& lAirlineClassQList =
    FacBom<AirlineClassList>::instance().create (lAirlineClassQListKey);
  // Link the AirlineClassList to the FareFeatures object
  FacBomManager::addToListAndMap (lFareFeaturesQ, lAirlineClassQList);
  FacBomManager::linkWithParent (lFareFeaturesQ, lAirlineClassQList);
  lAirlineClassQList.setFare (600);
  lAirlineClassQList.setYield (600);

  // Link the AirlineClassList to the YieldFeatures object
  FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassQList);
  // \todo (gsabatier): the following calls overrides the parent for
  //       lAirlineClassList. Check that it is what is actually wanted.
  FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassQList);

}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::
buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
    ) {

  // Clean the list
  ioTravelSolutionList.clear();

  //
  const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");


  // Add the segment date key to the travel solution
  TravelSolutionStruct lTS;
  lTS.addSegment (lBA9_SegmentDateKey);

  // Add the travel solution to the list
  ioTravelSolutionList.push_back (lTS);
}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::
buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {

  // Clean the list
  ioTravelSolutionList.clear();

  //
  const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");


  // Add the segment date key to the travel solution
  TravelSolutionStruct lTS1;
  lTS1.addSegment (lBA9_SegmentDateKey);

  // Fare option number 1
  const ClassCode_T lClassPathQ (CLASS_CODE_Q);
```

```
const Fare_T lFare900 (900);
const ChangeFees_T lChangeFee (CHANGE_FEES);
const NonRefundable_T isNonRefundable (NON_REFUNDABLE);
const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
const FareOptionStruct lFareOption1 (lClassPathQ, lFare900, lChangeFee,
                                     isNonRefundable, lSaturdayStay);

// Add (a copy of) the fare option
lTS1.addFareOption (lFareOption1);
//

// Map of class availabilities: set the availability for the Q
// booking class (the one corresponding to the fare option) to 8.
ClassAvailabilityMap_T lClassAvailabilityMap1;
const Availability_T lAvl1 (8);
bool hasInsertOfQBeenSuccessful = lClassAvailabilityMap1.
  insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl1)).second;
assert (hasInsertOfQBeenSuccessful == true);
// Add the map to the dedicated list held by the travel solution
lTS1.addClassAvailabilityMap (lClassAvailabilityMap1);

// Add the travel solution to the list
ioTravelSolutionList.push_back (lTS1);

//
const std::string lQF12_SegmentDateKey ("QF, 12, 2011-06-10, LHR, SYD, 20:45"
  );

// Add the segment date key to the travel solution
TravelSolutionStruct lTS2;
lTS2.addSegment (lQF12_SegmentDateKey);

// Fare option number 2
const ClassCode_T lClassPathY (CLASS_CODE_Y);
const Fare_T lFare1000 (1000);
const ChangeFees_T lNoChangeFee (NO_CHANGE_FEES);
const NonRefundable_T isRefundable (NO_NON_REFUNDABLE);
const FareOptionStruct lFareOption2 (lClassPathY, lFare1000, lNoChangeFee,
                                     isRefundable, lSaturdayStay);

// Map of class availabilities: set the availability for the Y
// booking class (the one corresponding to the fare option) to 9.
ClassAvailabilityMap_T lClassAvailabilityMap2;
const Availability_T lAvl2 (9);
const bool hasInsertOfYBeenSuccessful = lClassAvailabilityMap2.
  insert (ClassAvailabilityMap_T::value_type (lClassPathY, lAvl2)).second;
assert (hasInsertOfYBeenSuccessful == true);
// Add the map to the dedicated list held by the travel solution
lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

// Add (a copy of) the fare option
lTS2.addFareOption (lFareOption2);

// Fare option number 3
const Fare_T lFare920 (920);
const FareOptionStruct lFareOption3 (lClassPathQ, lFare920, lNoChangeFee,
                                     isNonRefundable, lSaturdayStay);

// Map of class availabilities: set the availability for the Q
// booking class (the one corresponding to the fare option) to 9.
hasInsertOfQBeenSuccessful = lClassAvailabilityMap2.
  insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl2)).second;
assert (hasInsertOfYBeenSuccessful == true);
// Add the map to the dedicated list held by the travel solution
lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

// Add (a copy of) the fare option
```

```
    lTS2.addFareOption (lFareOption3);

    // Add the travel solution to the list
    ioTravelSolutionList.push_back (lTS2);

  }

  // ////////////////////////////////////////////////////////////////////////
  BookingRequestStruct CmdBomManager::buildSampleBookingRequest() {
    // Origin
    const AirportCode_T lOrigin (AIRPORT_LHR);

    // Destination
    const AirportCode_T lDestination (AIRPORT_SYD);

    // Point of Sale (POS)
    const CityCode_T lPOS (POS_LHR);

    // Preferred departure date (10-JUN-2011)
    const Date_T lPreferredDepartureDate (2011, boost::gregorian::Jun, 10);

    // Preferred departure time (08:00)
    const Duration_T lPreferredDepartureTime (8, 0, 0);

    // Date of the request (15-MAY-2011)
    const Date_T lRequestDate (2011, boost::gregorian::May, 15);

    // Time of the request (10:00)
    const Duration_T lRequestTime (10, 0, 0);

    // Date-time of the request (made of the date and time above)
    const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

    // Preferred cabin (also named class of service sometimes)
    const CabinCode_T lPreferredCabin (CABIN_ECO);

    // Number of persons in the party
    const PartySize_T lPartySize (3);

    // Channel (direct/indirect, on-line/off-line)
    const ChannelLabel_T lChannel (CHANNEL_DN);

    // Type of the trip (one-way, inbound/outbound of a return trip)
    const TripType_T lTripType (TRIP_TYPE_INBOUND);

    // Duration of the stay (expressed as a number of days)
    const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

    // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
    const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

    // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
    const WTP_T lWTP (DEFAULT_WTP);

    // Value of time, for the customer (expressed in monetary unit per
    // unit of time, e.g., EUR/hour)
    const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

    // Restrictions
    const ChangeFees_T lChangeFees = false;
    const Disutility_T lChangeFeeDisutility = 30;
    const NonRefundable_T lNonRefundable = false;
    const Disutility_T lNonRefundableDisutility = 50;

    // Creation of the booking request structure
    BookingRequestStruct oBookingRequest (lOrigin, lDestination, lPOS,
                                          lPreferredDepartureDate,
```

```
                                        lRequestDateTime,
                                        lPreferredCabin,
                                        lPartySize, lChannel,
                                        lTripType, lStayDuration,
                                        lFrequentFlyerType,
                                        lPreferredDepartureTime,
                                        lWTP, lValueOfTime,
                                        lChangeFees, lChangeFeeDisutility,
                                        lNonRefundable,
                                        lNonRefundableDisutility);

  return oBookingRequest;
}

// ////////////////////////////////////////////////////////////////////
BookingRequestStruct CmdBomManager::buildSampleBookingRequestForCRS() {
  // Origin
  const AirportCode_T lOrigin (AIRPORT_SIN);

  // Destination
  const AirportCode_T lDestination (AIRPORT_BKK);

  // Point of Sale (POS)
  const CityCode_T lPOS (POS_SIN);

  // Preferred departure date (30-JAN-2010)
  const Date_T lPreferredDepartureDate (2010, boost::gregorian::Jan, 30);

  // Preferred departure time (10:00)
  const Duration_T lPreferredDepartureTime (10, 0, 0);

  // Date of the request (22-JAN-2010)
  const Date_T lRequestDate (2010, boost::gregorian::Jan, 22);

  // Time of the request (10:00)
  const Duration_T lRequestTime (10, 0, 0);

  // Date-time of the request (made of the date and time above)
  const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

  // Preferred cabin (also named class of service sometimes)
  const CabinCode_T lPreferredCabin (CABIN_ECO);

  // Number of persons in the party
  const PartySize_T lPartySize (3);

  // Channel (direct/indirect, on-line/off-line)
  const ChannelLabel_T lChannel (CHANNEL_IN);

  // Type of the trip (one-way, inbound/outbound of a return trip)
  const TripType_T lTripType (TRIP_TYPE_INBOUND);

  // Duration of the stay (expressed as a number of days)
  const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

  // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
  const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

  // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
  const WTP_T lWTP (DEFAULT_WTP);

  // Value of time, for the customer (expressed in monetary unit per
  // unit of time, e.g., EUR/hour)
  const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

  // Restrictions
  const ChangeFees_T lChangeFees = true;
```

```
  const Disutility_T lChangeFeeDisutility = 50;
  const NonRefundable_T lNonRefundable = true;
  const Disutility_T lNonRefundableDisutility = 50;

  // Creation of the booking request structure
  BookingRequestStruct oBookingRequest (lOrigin,
                                        lDestination,
                                        lPOS,
                                        lPreferredDepartureDate,
                                        lRequestDateTime,
                                        lPreferredCabin,
                                        lPartySize, lChannel,
                                        lTripType, lStayDuration,
                                        lFrequentFlyerType,
                                        lPreferredDepartureTime,
                                        lWTP, lValueOfTime,
                                        lChangeFees, lChangeFeeDisutility,
                                        lNonRefundable,
                                        lNonRefundableDisutility);

  return oBookingRequest;
}

// ////////////////////////////////////////////////////////////////////////
void CmdBomManager::
buildPartnershipsSampleInventoryAndRM (BomRoot& ioBomRoot) {

  // Step 0.1: Inventory level
  // Create an Inventory for SQ
  const AirlineCode_T lAirlineCodeSQ ("SQ");
  const InventoryKey lSQKey (lAirlineCodeSQ);
  Inventory& lSQInv = FacBom<Inventory>::instance().create (lSQKey);
  FacBomManager::addToListAndMap (ioBomRoot, lSQInv);
  FacBomManager::linkWithParent (ioBomRoot, lSQInv);

  // Add the airline feature object to the SQ inventory
  const AirlineFeatureKey lAirlineFeatureSQKey (lAirlineCodeSQ);
  AirlineFeature& lAirlineFeatureSQ =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureSQKey);
  FacBomManager::setAirlineFeature (lSQInv, lAirlineFeatureSQ);
  FacBomManager::linkWithParent (lSQInv, lAirlineFeatureSQ);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureSQ);

  // Create an Inventory for CX
  const AirlineCode_T lAirlineCodeCX ("CX");
  const InventoryKey lCXKey (lAirlineCodeCX);
  Inventory& lCXInv = FacBom<Inventory>::instance().create (lCXKey);
  FacBomManager::addToListAndMap (ioBomRoot, lCXInv);
  FacBomManager::linkWithParent (ioBomRoot, lCXInv);

  // Add the airline feature object to the CX inventory
  const AirlineFeatureKey lAirlineFeatureCXKey (lAirlineCodeCX);
  AirlineFeature& lAirlineFeatureCX =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureCXKey);
  FacBomManager::setAirlineFeature (lCXInv, lAirlineFeatureCX);
  FacBomManager::linkWithParent (lCXInv, lAirlineFeatureCX);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureCX);

  // ////// SQ ///////
  // Step 0.2: Flight-date level
  // Create a FlightDate (SQ11/08-MAR-2010) for SQ's Inventory
  FlightNumber_T lFlightNumber = 11;
  Date_T lDate (2010, 3, 8);
  FlightDateKey lFlightDateKey (lFlightNumber, lDate);
```

```
FlightDate& lSQ11_20100308_FD =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ11_20100308_FD);
FacBomManager::linkWithParent (lSQInv, lSQ11_20100308_FD);

// Create a (mkt) FlightDate (SQ1200/08-MAR-2010) for SQ's Inventory
FlightNumber_T lMktFlightNumber = 1200;
//lDate = Date_T (2010, 3, 8);
FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);

FlightDate& lSQ1200_20100308_FD =
  FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100308_FD);
FacBomManager::linkWithParent (lSQInv, lSQ1200_20100308_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());

// Step 0.3: Segment-date level
// Create a first SegmentDate (SIN-BKK) for SQ's Inventory
const AirportCode_T lSIN ("SIN");
const AirportCode_T lBKK ("BKK");
const DateOffset_T l1Day (1);
const DateOffset_T l2Days (2);
const Duration_T l0820 (8, 20, 0);
const Duration_T l1100 (11, 0, 0);
const Duration_T l0340 (3, 40, 0);
SegmentDateKey lSegmentDateKey (lSIN, lBKK);

SegmentDate& lSINBKKSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINBKKSegment);
FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINBKKSegment);

// Add the routing leg key to the SIN-BKK segment.
const std::string lSQSINRoutingLegStr = "SQ;11;2010-Mar-8;SIN";
lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

// Fill the SegmentDate content
lSINBKKSegment.setBoardingDate (lDate);
lSINBKKSegment.setOffDate (lDate);
lSINBKKSegment.setBoardingTime (l0820);
lSINBKKSegment.setOffTime (l1100);
lSINBKKSegment.setElapsedTime (l0340);

// Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
const AirportCode_T lHKG ("HKG");
const Duration_T l1200 (12, 0, 0);
const Duration_T l1540 (15, 40, 0);
const Duration_T l0240 (2, 40, 0);
SegmentDateKey lMktSegmentDateKey (lBKK, lHKG);

SegmentDate& lMktBKKHKGSegment =
  FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lSQ1200_20100308_FD, lMktBKKHKGSegment);
FacBomManager::linkWithParent (lSQ1200_20100308_FD, lMktBKKHKGSegment);

// Add the routing leg key CX;12;2010-Mar-8;BKK to the marketing
// SQ;1200;2010-Mar-8;BKK-HKG segment.
const std::string lCXBKKRoutingLegStr = "CX;12;2010-Mar-8;BKK";
lMktBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

// Fill the (mkt) SegmentDate content
lMktBKKHKGSegment.setBoardingDate (lDate);
lMktBKKHKGSegment.setOffDate (lDate);
lMktBKKHKGSegment.setBoardingTime (l1200);
lMktBKKHKGSegment.setOffTime (l1540);
```

```
lMktBKKHKGSegment.setElapsedTime (l0240);

// Step 0.4: Leg-date level
// Create a first LegDate (SIN) for SQ's Inventory
LegDateKey lLegDateKey (lSIN);

LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINLeg);
FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINLeg);

// Fill the LegDate content
lSINLeg.setOffPoint (lBKK);
lSINLeg.setBoardingDate (lDate);
lSINLeg.setOffDate (lDate);
lSINLeg.setBoardingTime (l0820);
lSINLeg.setOffTime (l1100);
lSINLeg.setElapsedTime (l0340);

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
const CabinCode_T lY ("Y");
SegmentCabinKey lYSegmentCabinKey (lY);

SegmentCabin& lSINBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);

// Create a SegmentCabin (Y) for the (mkt) Segment BKK-HKG of SQ's Inventory
SegmentCabin& lMktBKKHKGSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);
FacBomManager::linkWithParent (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);


// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
LegCabinKey lYLegCabinKey (lY);

LegCabin& lSINLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);

CabinCapacity_T lCapacity (100);
lSINLegYCabin.setCapacities (lCapacity);
lSINLegYCabin.setAvailabilityPool (lCapacity);


// Step 0.7: fare family level
// Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
const FamilyCode_T l1 ("EcoSaver");
FareFamilyKey l1FareFamilyKey (l1);

FareFamily& lSINBKKSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                               lSINBKKSegmentYCabin1Family);

// Create a FareFamily (1) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv

FareFamily& lMktBKKHKGSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1Family);
```

```
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
                               lMktBKKHKGSegmentYCabin1Family);

// Step 0.8: booking class level
// Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
BookingClassKey lYBookingClassKey (lY);

BookingClass& lSINBKKSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                               lSINBKKSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyYClass);

lSINBKKSegmentYCabin1FamilyYClass.setYield(700);

// Create a BookingClass (Y) for the (mkt) Segment BKK-HKG, cabin Y,
// fare family 1 on SQ's Inv
BookingClass& lMktBKKHKGSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin1Family,
                               lMktBKKHKGSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabin1FamilyYClass);

lMktBKKHKGSegmentYCabin1FamilyYClass.setYield(700);


// Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
// fare family 1 on SQ's Inv
const ClassCode_T lM ("M");
BookingClassKey lMBookingClassKey (lM);

BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                               lSINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                lSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lSINBKKSegment,
                                lSINBKKSegmentYCabin1FamilyMClass);

lSINBKKSegmentYCabin1FamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
// fare family 1 on SQ's Inv
BookingClass& lMktBKKHKGSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin1Family,
                               lMktBKKHKGSegmentYCabin1FamilyMClass);
```

```
FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                lMktBKKHKGSegmentYCabin1FamilyMClass);

lMktBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

/* ==========================================================================
  ========= */

// Step 1.0: O&D level
// Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK-SQ1200/08-MAR-2010/BKK-HKG)
// for SQ's Inventory
OnDString_T lSQSINBKKOnDStr = "SQ;11,2010-Mar-08;SIN,BKK";
OnDString_T lMktSQBKKHKGOnDStr = "SQ;1200,2010-Mar-08;BKK,HKG";
OnDStringList_T lOnDStringList;
lOnDStringList.push_back (lSQSINBKKOnDStr);
lOnDStringList.push_back (lMktSQBKKHKGOnDStr);

OnDDateKey lOnDDateKey (lOnDStringList);
OnDDate& lSQ_SINHKG_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lSQ_SINHKG_OnDDate);
FacBomManager::linkWithParent (lSQInv, lSQ_SINHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lSINBKKSegment);
FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lMktBKKHKGSegment);

// Add total forecast info for cabin Y.
const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
const WTP_T lWTP750 = 750.0;
const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
lSQ_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

// Add demand info (optional).
// 2 legs here, so 2 CabinClassPair to add in the list.
// Fist leg: cabin Y, class M.
CabinClassPair_T lCC_YM1 (lY,lM);
// Second leg: cabin Y, class M too.
CabinClassPair_T lCC_YM2 (lY,lM);
CabinClassPairList_T lCabinClassPairList;
lCabinClassPairList.push_back(lCC_YM1);
lCabinClassPairList.push_back(lCC_YM2);
const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
const Yield_T lYield850 = 850.0;
const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
  StdDev2);

CabinClassPair_T lCC_YY1 (lY,lY);
CabinClassPair_T lCC_YY2 (lY,lY);
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
lCabinClassPairList.push_back(lCC_YY2);
const MeanStdDevPair_T lMean10StdDev1 (10.0, 1.0);
const Yield_T lYield1200 = 1200.0;
const YieldDemandPair_T lYield1200Mean10StdDev1 (lYield1200,
                                                 lMean10StdDev1);
lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield1200Mean10StdDev1);

// Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK) for SQ's Inventory
lOnDStringList.clear();
lOnDStringList.push_back (lSQSINBKKOnDStr);
```

```
lOnDDateKey = OnDDateKey(lOnDStringList);
OnDDate& lSQ_SINBKK_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);

// Add total forecast info for cabin Y.
const WTP_T lWTP400 = 400.0;
const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

// Add demand info (optional).
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM1);
const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
const Yield_T lYield500 = 500.0;
const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                          lYield500Mean20StdDev1);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
const Yield_T lYield700 = 700.0;
const YieldDemandPair_T lYield700Mean20StdDev1 (lYield700, lMean10StdDev1 );
lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                          lYield700Mean20StdDev1);

/*****************************************************************************
  ***
// Create an O&D Date (SQ1200/08-MAR-2010/BKK-HKG) for SQ's Inventory
lFullKeyList.clear();
lFullKeyList.push_back (lMktSQBKKHKGFullKeyStr);

lOnDDateKey = OnDDateKey(lFullKeyList);
OnDDate& lMktSQ_BKKHKG_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lSQInv, lMktSQ_BKKHKG_OnDDate);
FacBomManager::linkWithParent (lSQInv, lMktSQ_BKKHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lMktSQ_BKKHKG_OnDDate, lMktBKKHKGSegment);

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM2);
// lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
  .0, 1.0);
*****************************************************************************
  ******/


// ////// CX ///////
// Step 0.2: Flight-date level
// Create a FlightDate (CX12/08-MAR-2010) for CX's Inventory
lFlightNumber = 12;
//lDate = Date_T (2010, 2, 8);
lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lCX12_20100308_FD =
  FacBom<FlightDate>::instance().create (lFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX12_20100308_FD);
```

```
FacBomManager::linkWithParent (lCXInv, lCX12_20100308_FD);

// Create a (mkt) FlightDate (CX1100/08-FEB-2010) for CX's Inventory
lFlightNumber = 1100;
//lDate = Date_T (2010, 2, 8);
lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);

FlightDate& lCX1100_20100308_FD =
  FacBom<FlightDate>::instance().create (lMktFlightDateKey);
FacBomManager::addToListAndMap (lCXInv, lCX1100_20100308_FD);
FacBomManager::linkWithParent (lCXInv, lCX1100_20100308_FD);

// Display the flight-date
// STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

// Step 0.3: Segment-date level
// Create a SegmentDate BKK-HKG for CX's Inventory

lSegmentDateKey = SegmentDateKey (lBKK, lHKG);

SegmentDate& lBKKHKGSegment =
  FacBom<SegmentDate>::instance().create (lSegmentDateKey);
FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKHKGSegment);
FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKHKGSegment);

// Add the routing leg key to the marketing BKK-HKG segment.
lBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

// Fill the SegmentDate content
lBKKHKGSegment.setBoardingDate (lDate);
lBKKHKGSegment.setOffDate (lDate);
lBKKHKGSegment.setBoardingTime (l1200);
lBKKHKGSegment.setOffTime (l1540);
lBKKHKGSegment.setElapsedTime (l0240);

// Create a second (mkt) SegmentDate (SIN-BKK) for CX's Inventory
lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);

SegmentDate& lMktSINBKKSegment =
  FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
FacBomManager::addToListAndMap (lCX1100_20100308_FD, lMktSINBKKSegment);
FacBomManager::linkWithParent (lCX1100_20100308_FD, lMktSINBKKSegment);

// Add the routing leg key SQ;11;2010-Mar-8;SIN to the marketing
// CX;1100;2010-Mar-8;SIN-BKK segment.
lMktSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

// Fill the (mkt) SegmentDate content
lMktSINBKKSegment.setBoardingDate (lDate);
lMktSINBKKSegment.setOffDate (lDate);
lMktSINBKKSegment.setBoardingTime (l0820);
lMktSINBKKSegment.setOffTime (l1100);
lMktSINBKKSegment.setElapsedTime (l0340);

// Step 0.4: Leg-date level
// Create a LegDate (BKK) for CX's Inventory
lLegDateKey = LegDateKey (lBKK);

LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKLeg);
FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKLeg);

// Fill the LegDate content
lBKKLeg.setOffPoint (lHKG);
lBKKLeg.setBoardingDate (lDate);
lBKKLeg.setOffDate (lDate);
lBKKLeg.setBoardingTime (l1200);
```

```
lBKKLeg.setOffTime (l1540);
lBKKLeg.setElapsedTime (l0240);

// Display the leg-date
// STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

// Step 0.5: segment-cabin level
// Create a SegmentCabin (Y) for the Segment BKK-HKG of CX's Inventory
SegmentCabin& lBKKHKGSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKHKGSegmentYCabin);
FacBomManager::linkWithParent (lBKKHKGSegment, lBKKHKGSegmentYCabin);

// Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
SegmentCabin& lMktSINBKKSegmentYCabin =
  FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
FacBomManager::addToListAndMap (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);

// Step 0.6: leg-cabin level
// Create a LegCabin (Y) for the Leg BKK-HKG on CX's Inventory
LegCabin& lBKKLegYCabin =
  FacBom<LegCabin>::instance().create (lYLegCabinKey);
FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);

lCapacity = CabinCapacity_T(100);
lBKKLegYCabin.setCapacities (lCapacity);
lBKKLegYCabin.setAvailabilityPool (lCapacity);

// Step 0.7: fare family level
// Create a fareFamily (1) for the Segment BKK-HKG, cabin Y on CX's Inv
FareFamily& lBKKHKGSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1Family);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
                               lBKKHKGSegmentYCabin1Family);

// Create a FareFamily (1) for the (mkt) Segment SIN-BKK, cabin Y on CX's Inv

FareFamily& lMktSINBKKSegmentYCabin1Family =
  FacBom<FareFamily>::instance().create (l1FareFamilyKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1Family);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin,
                               lMktSINBKKSegmentYCabin1Family);


// Step 0.8: booking class level
// Create a BookingClass (Y) for the
// Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
BookingClass& lBKKHKGSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                lBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                               lBKKHKGSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lBKKHKGSegment,
                                lBKKHKGSegmentYCabin1FamilyYClass);

lBKKHKGSegmentYCabin1FamilyYClass.setYield(700);

// Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
```

```
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabin1FamilyYClass =
  FacBom<BookingClass>::instance().create (lYBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                lMktSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                               lMktSINBKKSegmentYCabin1FamilyYClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1FamilyYClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabin1FamilyYClass);

lMktSINBKKSegmentYCabin1FamilyYClass.setYield(700);

//Create a BookingClass (M) for the
// Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
BookingClass& lBKKHKGSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                lBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                               lBKKHKGSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                lBKKHKGSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lBKKHKGSegment,
                                lBKKHKGSegmentYCabin1FamilyMClass);

lBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

// Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
// fare family 1 on CX's Inv
BookingClass& lMktSINBKKSegmentYCabin1FamilyMClass =
  FacBom<BookingClass>::instance().create (lMBookingClassKey);
FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                lMktSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                               lMktSINBKKSegmentYCabin1FamilyMClass);

FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                lMktSINBKKSegmentYCabin1FamilyMClass);
FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                lMktSINBKKSegmentYCabin1FamilyMClass);

lMktSINBKKSegmentYCabin1FamilyMClass.setYield(500);

/* ==========================================================================
  ========= */

// Step 1.0: O&D level
// Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK-CX12/08-MAR-2010/BKK-HKG) f
  or CX's Inventory
OnDString_T lMktCXSINBKKOnDStr = "CX;1100,2010-Mar-08;SIN,BKK";
OnDString_T lCXBKKHKGOnDStr = "CX;12,2010-Mar-08;BKK,HKG";
lOnDStringList.clear();
lOnDStringList.push_back (lMktCXSINBKKOnDStr);
lOnDStringList.push_back (lCXBKKHKGOnDStr);

lOnDDateKey = OnDDateKey(lOnDStringList);
OnDDate& lCX_SINHKG_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lCX_SINHKG_OnDDate);
FacBomManager::linkWithParent (lCXInv, lCX_SINHKG_OnDDate);

// Add the segments
```

```
FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lMktSINBKKSegment);
FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lBKKHKGSegment);

// Add total forecast info for cabin Y.
lCX_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM1);
lCabinClassPairList.push_back(lCC_YM2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield850Mean20StdDev2);

lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY1);
lCabinClassPairList.push_back(lCC_YY2);
lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield1200Mean10StdDev1);

/*****************************************************************************
  *******
// Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK) for CX's Inventory
lFullKeyList.clear();
lFullKeyList.push_back (lMktCXSINBKKFullKeyStr);

lOnDDateKey = OnDDateKey(lFullKeyList);
OnDDate& lMktCX_SINBKK_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lMktCX_SINBKK_OnDDate);
FacBomManager::linkWithParent (lCXInv, lMktCX_SINBKK_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lMktCX_SINBKK_OnDDate, lMktSINBKKSegment);

// Demand info is not added for purely marketed O&Ds
// Add demand info
// lCabinClassPairList.clear();
// lCabinClassPairList.push_back(lCC_YM1);
// lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20
  .0, 1.0);
*****************************************************************************
  ********/

// Create an O&D Date (CX12/08-FEB-2010/BKK-HKG) for CX's Inventory
lOnDStringList.clear();
lOnDStringList.push_back (lCXBKKHKGOnDStr);

lOnDDateKey = OnDDateKey(lOnDStringList);
OnDDate& lCX_BKKHKG_OnDDate =
  FacBom<OnDDate>::instance().create (lOnDDateKey);
// Link to the inventory
FacBomManager::addToListAndMap (lCXInv, lCX_BKKHKG_OnDDate);
FacBomManager::linkWithParent (lCXInv, lCX_BKKHKG_OnDDate);

// Add the segments
FacBomManager::addToListAndMap (lCX_BKKHKG_OnDDate, lBKKHKGSegment);

// Add total forecast info for cabin Y.
lCX_BKKHKG_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

// Add demand info
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YM2);
lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield500Mean20StdDev1);
```

```
lCabinClassPairList.clear();
lCabinClassPairList.push_back(lCC_YY2);
const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );
lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                         lYield700Mean10StdDev1);

/*==========================================================================
  =====
  ==========================================================================
  =====
  ==========================================================================
  =====*/
// Schedule:
// SQ:
// Step 1: flight period level
// Create a flight period for SQ11:
const DoWStruct lDoWSrtuct ("1111111");
const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 8);
const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 9);
const DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
const PeriodStruct lPeriodStruct (lDatePeriod,lDoWSrtuct);

lFlightNumber = FlightNumber_T (11);

FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);

FlightPeriod& lSQ11FlightPeriod =
  FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
FacBomManager::addToListAndMap (lSQInv, lSQ11FlightPeriod);
FacBomManager::linkWithParent (lSQInv, lSQ11FlightPeriod);

// Step 2: segment period level
// Create a segment period for SIN-BKK:

SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);

SegmentPeriod& lSINBKKSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
FacBomManager::addToListAndMap (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
FacBomManager::linkWithParent (lSQ11FlightPeriod, lSINBKKSegmentPeriod);

lSINBKKSegmentPeriod.setBoardingTime (l0820);
lSINBKKSegmentPeriod.setOffTime (l1100);
lSINBKKSegmentPeriod.setElapsedTime (l0340);
ClassList_String_T lYM ("YM");
lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYM);

// CX:
// Step 1: flight period level
// Create a flight period for CX12:
lFlightNumber = FlightNumber_T (12);

lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);

FlightPeriod& lCX12FlightPeriod =
  FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
FacBomManager::addToListAndMap (lCXInv, lCX12FlightPeriod);
FacBomManager::linkWithParent (lCXInv, lCX12FlightPeriod);

// Step 2: segment period level
// Create a segment period for BKK-HKG:

lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);

SegmentPeriod& lBKKHKGSegmentPeriod =
  FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
FacBomManager::addToListAndMap (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
```

```
    FacBomManager::linkWithParent (lCX12FlightPeriod, lBKKHKGSegmentPeriod);

    lBKKHKGSegmentPeriod.setBoardingTime (l1200);
    lBKKHKGSegmentPeriod.setOffTime (l1540);
    lBKKHKGSegmentPeriod.setElapsedTime (l0240);
    lBKKHKGSegmentPeriod.addCabinBookingClassList (lY,lYM);

  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildPartnershipsSamplePricing (BomRoot& ioBomRoot) {


    /*=========================================================================
      ========*/
    // First airport pair SIN-BKK.
    // Set the airport-pair primary key.
    AirportPairKey lAirportPairKey ("SIN", "BKK");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lSINBKKAirportPair =
      FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSINBKKAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lSINBKKAirportPair);

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 01);
    const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 31);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lSINBKKDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lSINBKKAirportPair, lSINBKKDatePeriod);
    FacBomManager::linkWithParent (lSINBKKAirportPair, lSINBKKDatePeriod);

    // Set the point-of-sale-channel primary key.
    PosChannelKey lPosChannelKey ("SIN","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lSINPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINBKKDatePeriod, lSINPosChannel);
    FacBomManager::linkWithParent (lSINBKKDatePeriod, lSINPosChannel);

    // Set the fare time-period primary key.
    const Time_T lTimeRangeStart (0, 0, 0);
    const Time_T lTimeRangeEnd (23, 0, 0);
    const TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
                                            lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINBKKFareTimePeriod =
      FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
    FacBomManager::addToListAndMap (lSINPosChannel, lSINBKKFareTimePeriod);
    FacBomManager::linkWithParent (lSINPosChannel, lSINBKKFareTimePeriod);

    // Generate the FareRule
    const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ONE_WAY,
                                            NO_ADVANCE_PURCHASE,
                                            SATURDAY_STAY,
                                            CHANGE_FEES,
                                            NON_REFUNDABLE,
                                            NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
```

```
FareFeatures& lSINBKKFareFeatures =
  FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lSINBKKFareTimePeriod, lSINBKKFareFeatures);
FacBomManager::linkWithParent (lSINBKKFareTimePeriod, lSINBKKFareFeatures);

// Generate Segment Features and link them to their FareRule.
AirlineCodeList_T lSQAirlineCodeList;
lSQAirlineCodeList.push_back ("SQ");

ClassList_StringList_T lYClassCodeList;
lYClassCodeList.push_back ("Y");
const AirlineClassListKey lSQAirlineYClassListKey (lSQAirlineCodeList,
                                                   lYClassCodeList);

ClassList_StringList_T lMClassCodeList;
lMClassCodeList.push_back ("M");
const AirlineClassListKey lSQAirlineMClassListKey (lSQAirlineCodeList,
                                                   lMClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lSQAirlineYClassList =
  FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
lSQAirlineYClassList.setFare(700);
FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineYClassList);
FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);

AirlineClassList& lSQAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
lSQAirlineMClassList.setFare(500);
FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineMClassList);
FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineMClassList);

/*==========================================================================
  ========*/
// Second airport pair BKK-HKG.
// Set the airport-pair primary key.
lAirportPairKey = AirportPairKey ("BKK", "HKG");

// Create the AirportPairKey object and link it to the ioBomRoot object.
AirportPair& lBKKHKGAirportPair =
  FacBom<AirportPair>::instance().create (lAirportPairKey);
FacBomManager::addToListAndMap (ioBomRoot, lBKKHKGAirportPair);
FacBomManager::linkWithParent (ioBomRoot, lBKKHKGAirportPair);

// Set the fare date-period primary key.
// Use the same as previously.

// Create the DatePeriodKey object and link it to the PosChannel object.
DatePeriod& lBKKHKGDatePeriod =
  FacBom<DatePeriod>::instance().create (lDatePeriodKey);
FacBomManager::addToListAndMap (lBKKHKGAirportPair, lBKKHKGDatePeriod);
FacBomManager::linkWithParent (lBKKHKGAirportPair, lBKKHKGDatePeriod);

// Set the point-of-sale-channel primary key.
lPosChannelKey  = PosChannelKey("BKK","IN");

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lBKKPosChannel =
  FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lBKKPosChannel);
FacBomManager::linkWithParent (lBKKHKGDatePeriod, lBKKPosChannel);

// Set the fare time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lBKKHKGFareTimePeriod =
```

```
  FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGFareTimePeriod);
FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGFareTimePeriod);

// Generate the FareRule
// Use the same key as previously.

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lBKKHKGFareFeatures =
  FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lBKKHKGFareTimePeriod, lBKKHKGFareFeatures);
FacBomManager::linkWithParent (lBKKHKGFareTimePeriod, lBKKHKGFareFeatures);

// Generate Segment Features and link them to their FareRule.
AirlineCodeList_T lCXAirlineCodeList;
lCXAirlineCodeList.push_back ("CX");

const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
                                                   lYClassCodeList);

const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
                                                   lMClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lCXAirlineYClassList =
  FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
lCXAirlineYClassList.setFare(700);
FacBomManager::addToListAndMap (lBKKHKGFareFeatures, lCXAirlineYClassList);
FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineYClassList);

AirlineClassList& lCXAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
lCXAirlineMClassList.setFare(500);
FacBomManager::addToListAndMap (lBKKHKGFareFeatures, lCXAirlineMClassList);
FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineMClassList);

/*==========================================================================
  ========*/
// Third airport pair SIN-HKG.
// Set the airport-pair primary key.
lAirportPairKey = AirportPairKey ("SIN", "HKG");

// Create the AirportPairKey object and link it to the ioBomRoot object.
AirportPair& lSINHKGAirportPair =
  FacBom<AirportPair>::instance().create (lAirportPairKey);
FacBomManager::addToListAndMap (ioBomRoot, lSINHKGAirportPair);
FacBomManager::linkWithParent (ioBomRoot, lSINHKGAirportPair);

// Set the fare date-period primary key.
// Use the same as previously.

// Create the DatePeriodKey object and link it to the PosChannel object.
DatePeriod& lSINHKGDatePeriod =
  FacBom<DatePeriod>::instance().create (lDatePeriodKey);
FacBomManager::addToListAndMap (lSINHKGAirportPair, lSINHKGDatePeriod);
FacBomManager::linkWithParent (lSINHKGAirportPair, lSINHKGDatePeriod);

// Set the point-of-sale-channel primary key.
lPosChannelKey = PosChannelKey("SIN","IN");

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lOnDSINPosChannel =
  FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lSINHKGDatePeriod, lOnDSINPosChannel);
FacBomManager::linkWithParent (lSINHKGDatePeriod, lOnDSINPosChannel);

// Set the fare time-period primary key.
```

```
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINHKGFareTimePeriod =
  FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
FacBomManager::addToListAndMap (lOnDSINPosChannel, lSINHKGFareTimePeriod);
FacBomManager::linkWithParent (lOnDSINPosChannel, lSINHKGFareTimePeriod);

// Generate the FareRule
// Use the same key as previously.

// Create the FareFeaturesKey and link it to the TimePeriod object.
FareFeatures& lSINHKGFareFeatures =
  FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
FacBomManager::addToListAndMap (lSINHKGFareTimePeriod, lSINHKGFareFeatures);
FacBomManager::linkWithParent (lSINHKGFareTimePeriod, lSINHKGFareFeatures);

// Generate Segment Features and link them to their FareRule.
AirlineCodeList_T lSQ_CXAirlineCodeList;
lSQ_CXAirlineCodeList.push_back ("SQ");
lSQ_CXAirlineCodeList.push_back ("CX");

ClassList_StringList_T lY_YClassCodeList;
lY_YClassCodeList.push_back ("Y");
lY_YClassCodeList.push_back ("Y");
const AirlineClassListKey lSQ_CXAirlineYClassListKey (lSQ_CXAirlineCodeList,
                                                      lY_YClassCodeList);

ClassList_StringList_T lM_MClassCodeList;
lM_MClassCodeList.push_back ("M");
lM_MClassCodeList.push_back ("M");
const AirlineClassListKey lSQ_CXAirlineMClassListKey (lSQ_CXAirlineCodeList,
                                                      lM_MClassCodeList);

// Create the AirlineClassListKey and link it to the FareFeatures object.
AirlineClassList& lSQ_CXAirlineYClassList =
  FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
lSQ_CXAirlineYClassList.setFare(1200);
FacBomManager::addToListAndMap (lSINHKGFareFeatures,
                                lSQ_CXAirlineYClassList);
FacBomManager::linkWithParent (lSINHKGFareFeatures,
                               lSQ_CXAirlineYClassList);

AirlineClassList& lSQ_CXAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
lSQ_CXAirlineMClassList.setFare(850);
FacBomManager::addToListAndMap (lSINHKGFareFeatures,
                                lSQ_CXAirlineMClassList);
FacBomManager::linkWithParent (lSINHKGFareFeatures,
                               lSQ_CXAirlineMClassList);



/*==========================================================================
  ========*/

// Use the same airport pair, and date period for adding SQ SIN-BKK yields.

// Set the point-of-sale-channel primary key.
lPosChannelKey = PosChannelKey(DEFAULT_POS, DEFAULT_CHANNEL);

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_SINBKKPosChannel =
  FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
FacBomManager::linkWithParent (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
```

```
// Set the yield time-period primary key.
const TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
                                         lTimeRangeEnd);

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINBKKYieldTimePeriod =
  FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_SINBKKPosChannel,
                                lSINBKKYieldTimePeriod);
FacBomManager::linkWithParent (lRAC_SINBKKPosChannel,
                               lSINBKKYieldTimePeriod);

// Generate the YieldRule
const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
                                          CABIN_Y);

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lSINBKKYieldFeatures =
  FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lSINBKKYieldTimePeriod,
                                lSINBKKYieldFeatures);
FacBomManager::linkWithParent (lSINBKKYieldTimePeriod,
                               lSINBKKYieldFeatures);

// Generate Segment Features and link them to their YieldRule.
// Use the same key as previously.

// Create the AirlineClassListKey and link it to the YieldFeatures object.
AirlineClassList& lRAC_SQAirlineYClassList =
  FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
lRAC_SQAirlineYClassList.setYield(700);
FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
                                lRAC_SQAirlineYClassList);
FacBomManager::linkWithParent (lSINBKKYieldFeatures,
                               lRAC_SQAirlineYClassList);

AirlineClassList& lRAC_SQAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
lRAC_SQAirlineMClassList.setYield(500);
FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
                                lRAC_SQAirlineMClassList);
FacBomManager::linkWithParent (lSINBKKYieldFeatures,
                               lRAC_SQAirlineMClassList);

/*==========================================================================
  ========*/

// Use the same airport pair, and date period for adding CX BKK-HKG yields.

// Set the point-of-sale-channel primary key.
// Use the same as previously.

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_BKKHKGPosChannel =
  FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
FacBomManager::linkWithParent (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);

// Set the yield time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lBKKHKGYieldTimePeriod =
  FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_BKKHKGPosChannel,
                                lBKKHKGYieldTimePeriod);
FacBomManager::linkWithParent (lRAC_BKKHKGPosChannel,
```

```
                                lBKKHKGYieldTimePeriod);

// Generate the YieldRule
// Use the same key as previously.

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lBKKHKGYieldFeatures =
  FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lBKKHKGYieldTimePeriod,
                                 lBKKHKGYieldFeatures);
FacBomManager::linkWithParent (lBKKHKGYieldTimePeriod,
                                lBKKHKGYieldFeatures);

// Generate Segment Features and link them to their YieldRule.
// Use the same key as previously.

// Create the AirlineClassListKey and link it to the YieldFeatures object.
AirlineClassList& lRAC_CXAirlineYClassList =
  FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
lRAC_CXAirlineYClassList.setYield(700);
FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
                                 lRAC_CXAirlineYClassList);
FacBomManager::linkWithParent (lBKKHKGYieldFeatures,
                                lRAC_CXAirlineYClassList);

AirlineClassList& lRAC_CXAirlineMClassList =
  FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
lRAC_CXAirlineMClassList.setYield(500);
FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
                                 lRAC_CXAirlineMClassList);
FacBomManager::linkWithParent (lBKKHKGYieldFeatures,
                                lRAC_CXAirlineMClassList);

/*==========================================================================
  ========*/

// Use the same airport pair, and date period for SQ-CX SIN-HKG

// Set the point-of-sale-channel primary key.
// Use the same as previously.

// Create the PositionKey object and link it to the AirportPair object.
PosChannel& lRAC_SINHKGChannel =
  FacBom<PosChannel>::instance().create (lPosChannelKey);
FacBomManager::addToListAndMap (lSINHKGDatePeriod, lRAC_SINHKGChannel);
FacBomManager::linkWithParent (lSINHKGDatePeriod, lRAC_SINHKGChannel);

// Set the yield time-period primary key.
// Use the same as previously.

// Create the TimePeriodKey and link it to the DatePeriod object.
TimePeriod& lSINHKGYieldTimePeriod =
  FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
FacBomManager::addToListAndMap (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);
FacBomManager::linkWithParent (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);


// Generate the YieldRule
// Use the same key as previously.

// Create the YieldFeaturesKey and link it to the TimePeriod object.
YieldFeatures& lSINHKGYieldFeatures =
  FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
FacBomManager::addToListAndMap (lSINHKGYieldTimePeriod,
                                 lSINHKGYieldFeatures);
FacBomManager::linkWithParent (lSINHKGYieldTimePeriod,
                                lSINHKGYieldFeatures);
```

```
    // Generate Segment Features and link them to their YieldRule.
    // Use the same key as previously

    // Create the AirlineClassListKey and link it to the YieldFeatures object.
    AirlineClassList& lRAC_SQ_CXAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
    lRAC_SQ_CXAirlineYClassList.setYield(1200);
    FacBomManager::addToListAndMap (lSINHKGYieldFeatures,
                                    lRAC_SQ_CXAirlineYClassList);
    FacBomManager::linkWithParent (lSINHKGYieldFeatures,
                                   lRAC_SQ_CXAirlineYClassList);

    AirlineClassList& lRAC_SQ_CXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
    lRAC_SQ_CXAirlineMClassList.setYield(850);
    FacBomManager::addToListAndMap (lSINHKGYieldFeatures,
                                    lRAC_SQ_CXAirlineMClassList);
    FacBomManager::linkWithParent (lSINHKGYieldFeatures,
                                   lRAC_SQ_CXAirlineMClassList);

  }

}

/*!


 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacCloneBom.hpp>
#include <stdair/command/CmdCloneBomManager.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/BomRetriever.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  void CmdCloneBomManager::cloneBomRoot (const BomRoot& iBomRoot,
                                         BomRoot& ioCloneBomRoot) {

    // Check whether there are Inventory objects
    const bool hasInventoryList = BomManager::hasList<Inventory> (iBomRoot);
    if (hasInventoryList == true) {

      // Browse the inventories
      const InventoryList_T& lInventoryList =
        BomManager::getList<Inventory> (iBomRoot);
      for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
           itInv != lInventoryList.end(); ++itInv) {
        const Inventory* lInv_ptr = *itInv;
        assert (lInv_ptr != NULL);

        // Clone the current inventory
        Inventory& lCloneInventory = cloneInventory (*lInv_ptr, ioCloneBomRoot);
        FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneInventory);
        FacBomManager::linkWithParent (ioCloneBomRoot, lCloneInventory);
      }
    }

    // Check whether there are Airport Pair objects
```

```
  const bool hastAirportPairList =
    BomManager::hasList<AirportPair> (iBomRoot);
  if (hastAirportPairList == true) {

    // Browse the airport pairs
    const AirportPairList_T& lAirportPairList =
      BomManager::getList<AirportPair> (iBomRoot);
    for (AirportPairList_T::const_iterator itAirportPair =
           lAirportPairList.begin();
         itAirportPair != lAirportPairList.end(); ++itAirportPair) {
      const AirportPair* lAirportPair_ptr = *itAirportPair;
      assert (lAirportPair_ptr != NULL);

      // Clone the current airport pair
      AirportPair& lCloneAirportPair = cloneAirportPair (*lAirportPair_ptr);
      FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirportPair);
      FacBomManager::linkWithParent (ioCloneBomRoot, lCloneAirportPair);
    }
  }
}

// ////////////////////////////////////////////////////////////////////
Inventory& CmdCloneBomManager::cloneInventory (const Inventory& iInventory,
                                               BomRoot& ioCloneBomRoot) {

  Inventory& lCloneInventory =
    FacCloneBom<Inventory>::instance().clone (iInventory);

  // Check whether there are FlightDate objects
  const bool hasFlighDateList = BomManager::hasList<FlightDate> (iInventory);
  if (hasFlighDateList == true) {
    // Browse the flight-dates
    const FlightDateList_T& lFlightDateList =
      BomManager::getList<FlightDate> (iInventory);
    for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
         itFD != lFlightDateList.end(); ++itFD) {
      const FlightDate* lFD_ptr = *itFD;
      assert (lFD_ptr != NULL);

      // Clone the current flight-date
      FlightDate& lCloneFD = cloneFlightDate (*lFD_ptr);
      FacBomManager::addToListAndMap (lCloneInventory, lCloneFD);
      FacBomManager::linkWithParent (lCloneInventory, lCloneFD);
    }
  }

  // Check if the inventory contains a list of partners
  const bool hasPartnerList = BomManager::hasList<Inventory> (iInventory);
  if (hasPartnerList == true) {

    // Browse the partner's inventories
    const InventoryList_T& lPartnerInventoryList =
      BomManager::getList<Inventory> (iInventory);

    for (InventoryList_T::const_iterator itInv =
           lPartnerInventoryList.begin();
         itInv != lPartnerInventoryList.end(); ++itInv) {
      const Inventory* lInv_ptr = *itInv;
      assert (lInv_ptr != NULL);

      // Clone the current partnership inventory
      Inventory& lClonePartnerInventory = cloneInventory (*lInv_ptr,
                                                          ioCloneBomRoot);
      FacBomManager::addToListAndMap (lCloneInventory,
                                      lClonePartnerInventory);
      FacBomManager::linkWithParent (lCloneInventory,
                                     lClonePartnerInventory);
```

```
      }
    }

    // Check whether there are O&D date objects
    const bool hasOnDList = BomManager::hasList<OnDDate> (iInventory);
    if (hasOnDList == true){

      //Browse the O&Ds
      const OnDDateList_T& lOnDDateList =
        BomManager::getList<OnDDate> (iInventory);

      for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
           itOnD != lOnDDateList.end(); ++itOnD) {
        const OnDDate* lOnDDate_ptr = *itOnD;
        assert (lOnDDate_ptr != NULL);

        // Clone the current O&D date
        OnDDate& lCloneOnDDate = cloneOnDDate (*lOnDDate_ptr);
        FacBomManager::addToListAndMap (lCloneInventory, lCloneOnDDate);
        FacBomManager::linkWithParent (lCloneInventory, lCloneOnDDate);
      }
    }

    // Check whether there are Flight Period objects
    const bool hasFlightPeriodList =
      BomManager::hasList<FlightPeriod> (iInventory);
    if (hasFlightPeriodList == true) {

      // Browse the flight-periods
      const FlightPeriodList_T& lFlightPeriodList =
        BomManager::getList<FlightPeriod> (iInventory);
      for (FlightPeriodList_T::const_iterator itFlightPeriod =
             lFlightPeriodList.begin();
           itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
        const FlightPeriod* lFlightPeriod_ptr = *itFlightPeriod;
        assert (lFlightPeriod_ptr != NULL);

        // Clone the current flight period
        FlightPeriod& lCloneFlightPeriod = cloneFlightPeriod (*lFlightPeriod_ptr)
      ;
        FacBomManager::addToListAndMap (lCloneInventory, lCloneFlightPeriod);
        FacBomManager::linkWithParent (lCloneInventory, lCloneFlightPeriod);
      }
    }

    // Check whether there is an airline feature object
    const AirlineFeature* lAirlineFeature_ptr =
      BomManager::getObjectPtr<AirlineFeature,Inventory> (iInventory,
                                                          iInventory.getAirlineCo
      de());
    if (lAirlineFeature_ptr != NULL) {
      // Clone the current airline feature object
      AirlineFeature& lCloneAirlineFeature =
        cloneAirlineFeature (*lAirlineFeature_ptr);
      FacBomManager::setAirlineFeature (lCloneInventory, lCloneAirlineFeature);
      FacBomManager::linkWithParent (lCloneInventory, lCloneAirlineFeature);
      // Link the airline feature object with the top of the BOM tree
      FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirlineFeature);
    }

    return lCloneInventory;
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineFeature& CmdCloneBomManager::
  cloneAirlineFeature (const AirlineFeature& iAirlineFeature) {
```

```
  AirlineFeature& lCloneAirlineFeature =
    FacCloneBom<AirlineFeature>::instance().clone (iAirlineFeature);

  return lCloneAirlineFeature;
}


// ////////////////////////////////////////////////////////////////////
OnDDate& CmdCloneBomManager::cloneOnDDate (const OnDDate& iOnDDate) {

  OnDDate& lCloneOnDDate =
    FacCloneBom<OnDDate>::instance().clone (iOnDDate);

  return lCloneOnDDate;
}

// ////////////////////////////////////////////////////////////////////
FlightDate& CmdCloneBomManager::
cloneFlightDate (const FlightDate& iFlightDate) {

  FlightDate& lCloneFlightDate =
    FacCloneBom<FlightDate>::instance().clone (iFlightDate);

  // Check whether there are LegDate objects
  const bool hasLegDateList = BomManager::hasList<LegDate> (iFlightDate);
  if (hasLegDateList == true) {

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
      BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
      const LegDate* lLD_ptr = *itLD;
      assert (lLD_ptr != NULL);

      // Clone the current leg-date
      LegDate& lCloneLegDate = cloneLegDate (*lLD_ptr);
      FacBomManager::addToListAndMap (lCloneFlightDate, lCloneLegDate);
      FacBomManager::linkWithParent (lCloneFlightDate, lCloneLegDate);
    }
  }

  // Check whether there are SegmentDate objects
  const bool hasSegmentDateList =
    BomManager::hasList<SegmentDate> (iFlightDate);
  if (hasSegmentDateList == true) {

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

      // Clone the current segment-date
      SegmentDate& lCloneSegmentDate = cloneSegmentDate (*lSD_ptr);
      FacBomManager::addToListAndMap (lCloneFlightDate, lCloneSegmentDate);
      FacBomManager::linkWithParent (lCloneFlightDate, lCloneSegmentDate);

    }
  }

  return lCloneFlightDate;
}

// ////////////////////////////////////////////////////////////////////
```

```
LegDate& CmdCloneBomManager::cloneLegDate (const LegDate& iLegDate) {

  LegDate& lCloneLegDate =
    FacCloneBom<LegDate>::instance().clone (iLegDate);

  // Check whether there are LegCabin objects
  const bool hasLegCabinList = BomManager::hasList<LegCabin> (iLegDate);
  if (hasLegCabinList == true) {
    // Browse the leg-cabins
    const LegCabinList_T& lLegCabinList =
      BomManager::getList<LegCabin> (iLegDate);
    for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
         itLC != lLegCabinList.end(); ++itLC) {
      const LegCabin* lLC_ptr = *itLC;
      assert (lLC_ptr != NULL);

      // Clone the current leg-cabin
      LegCabin& lCloneLegCabin = cloneLegCabin (*lLC_ptr);
      FacBomManager::addToListAndMap (lCloneLegDate, lCloneLegCabin);
      FacBomManager::linkWithParent (lCloneLegDate, lCloneLegCabin);
    }
  }

  return lCloneLegDate;
}

// ////////////////////////////////////////////////////////////////////
LegCabin& CmdCloneBomManager::cloneLegCabin (const LegCabin& iLegCabin) {

  LegCabin& lCloneLegCabin =
    FacCloneBom<LegCabin>::instance().clone (iLegCabin);

  // Check whether there are Bucket objects
  const bool hasBucketList = BomManager::hasList<Bucket> (iLegCabin);
  if (hasBucketList == true) {
    // Browse the buckets
    const BucketList_T& lBucketList =
      BomManager::getList<Bucket> (iLegCabin);
    for (BucketList_T::const_iterator itBucket = lBucketList.begin();
         itBucket != lBucketList.end(); ++itBucket) {
      const Bucket* lBucket_ptr = *itBucket;
      assert (lBucket_ptr != NULL);

      // Clone the current bucket
      Bucket& lCloneBucket = cloneBucket (*lBucket_ptr);
      FacBomManager::addToListAndMap (lCloneLegCabin, lCloneBucket);
      FacBomManager::linkWithParent (lCloneLegCabin, lCloneBucket);
    }
  }

  return lCloneLegCabin;
}

// ////////////////////////////////////////////////////////////////////
Bucket& CmdCloneBomManager::cloneBucket (const Bucket& iBucket) {

  Bucket& lCloneBucket =
    FacCloneBom<Bucket>::instance().clone (iBucket);

  return lCloneBucket;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate& CmdCloneBomManager::
cloneSegmentDate (const SegmentDate& iSegmentDate) {

  SegmentDate& lCloneSegmentDate =
```

```
      FacCloneBom<SegmentDate>::instance().clone (iSegmentDate);

    // Check whether there are SegmentCabin objects
    const bool hasSegmentCabinList =
      BomManager::hasList<SegmentCabin> (iSegmentDate);
    if (hasSegmentCabinList == true) {
      // Browse the segment-cabins
      const SegmentCabinList_T& lSegmentCabinList =
        BomManager::getList<SegmentCabin> (iSegmentDate);
      for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
           itSC != lSegmentCabinList.end(); ++itSC) {
        const SegmentCabin* lSC_ptr = *itSC;
        assert (lSC_ptr != NULL);

        // Clone the current segment-cabin
        SegmentCabin& lCloneSegmentCabin = cloneSegmentCabin (*lSC_ptr);
        FacBomManager::addToListAndMap (lCloneSegmentDate, lCloneSegmentCabin);
        FacBomManager::linkWithParent (lCloneSegmentDate, lCloneSegmentCabin);

        linkBookingClassesWithSegment (lCloneSegmentDate,
                                       lCloneSegmentCabin);

      }
    }
    return lCloneSegmentDate;
  }

  // ////////////////////////////////////////////////////////////////////
  void CmdCloneBomManager::
  linkBookingClassesWithSegment (SegmentDate& iCloneSegmentDate,
                                 SegmentCabin& iCloneSegmentCabin) {

    // Browse the fare families to link the booking-classes to the
    // segment-cabin and to the segment-date
    const bool hasFareFamilyList =
      BomManager::hasList<FareFamily> (iCloneSegmentCabin);
    if (hasFareFamilyList == true) {
      const FareFamilyList_T& lCloneFFList =
        BomManager::getList<FareFamily> (iCloneSegmentCabin);
      for (FareFamilyList_T::const_iterator itCloneFF = lCloneFFList.begin();
           itCloneFF != lCloneFFList.end(); ++itCloneFF) {
        const FareFamily* lCloneFF_ptr = *itCloneFF;
        assert (lCloneFF_ptr != NULL);

        // Browse the list of booking classes
        const bool hasBookingClasslist =
          BomManager::hasList<BookingClass> (*lCloneFF_ptr);
        if (hasBookingClasslist == true) {
          const BookingClassList_T& lCloneBCList =
            BomManager::getList<BookingClass> (*lCloneFF_ptr);
          for (BookingClassList_T::const_iterator itCloneBC =
                 lCloneBCList.begin();
               itCloneBC != lCloneBCList.end(); ++itCloneBC) {
            const BookingClass* lCloneBC_ptr = *itCloneBC;
            assert (lCloneBC_ptr != NULL);

            // Link the booking-class to the segment-cabin
            stdair::FacBomManager::addToListAndMap (iCloneSegmentCabin,
                                                    *lCloneBC_ptr);

            // Link the booking-class to the segment-date
            stdair::FacBomManager::addToListAndMap (iCloneSegmentDate,
                                                    *lCloneBC_ptr);
          }
        }
      }
    }
```

```
}

// ////////////////////////////////////////////////////////////////////
SegmentCabin& CmdCloneBomManager::
cloneSegmentCabin (const SegmentCabin& iSegmentCabin) {

  SegmentCabin& lCloneSegmentCabin =
    FacCloneBom<SegmentCabin>::instance().clone (iSegmentCabin);

  // Check whether there are fare family objects
  const bool hasFareFamilyList =
    BomManager::hasList<FareFamily> (iSegmentCabin);
  if (hasFareFamilyList == true) {
    // Browse the fare families
    const FareFamilyList_T& lFareFamilyList =
      BomManager::getList<FareFamily> (iSegmentCabin);
    for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
         itFF != lFareFamilyList.end(); ++itFF) {
      const FareFamily* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Clone the current fare-family
      FareFamily& lCloneFareFamily = cloneFareFamily (*lFF_ptr);
      FacBomManager::addToListAndMap (lCloneSegmentCabin, lCloneFareFamily);
      FacBomManager::linkWithParent (lCloneSegmentCabin, lCloneFareFamily);
    }
  }

  return lCloneSegmentCabin;
}

// ////////////////////////////////////////////////////////////////////
FareFamily& CmdCloneBomManager::
cloneFareFamily (const FareFamily& iFareFamily) {
  FareFamily& lCloneFareFamily =
    FacCloneBom<FareFamily>::instance().clone (iFareFamily);

  // Check whether there are booking classes objects
  const bool hasBookingClassList =
    BomManager::hasList<BookingClass> (iFareFamily);
  if (hasBookingClassList == true) {
    // Browse the list of booking classes
    const BookingClassList_T& lBookingClassList =
      BomManager::getList<BookingClass> (iFareFamily);
    for (BookingClassList_T::const_iterator itBookingClass =
           lBookingClassList.begin();
         itBookingClass != lBookingClassList.end(); ++itBookingClass) {
      const BookingClass* lBC_ptr = *itBookingClass;
      assert (lBC_ptr != NULL);

      // Clone the current booking class
      BookingClass& lCloneBookingClass = cloneBookingClass (*lBC_ptr);
      FacBomManager::addToListAndMap (lCloneFareFamily, lCloneBookingClass);
      FacBomManager::linkWithParent (lCloneFareFamily, lCloneBookingClass);
    }
  }

  return lCloneFareFamily;
}

// ////////////////////////////////////////////////////////////////////
BookingClass& CmdCloneBomManager::
cloneBookingClass (const BookingClass& iBookingClass) {

  BookingClass& lCloneBookingClass =
    FacCloneBom<BookingClass>::instance().clone (iBookingClass);
```

```
  return lCloneBookingClass;
}

// ////////////////////////////////////////////////////////////////////
AirportPair& CmdCloneBomManager::
cloneAirportPair (const AirportPair& iAirportPair) {

  AirportPair& lCloneAirportPair =
    FacCloneBom<AirportPair>::instance().clone (iAirportPair);

  // Check whether there are date-period objects
  const bool hasDatePeriodList =
    BomManager::hasList<DatePeriod> (iAirportPair);
  if (hasDatePeriodList == true) {
    // Browse the date-periods
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (iAirportPair);
    for (DatePeriodList_T::const_iterator itDatePeriod =
           lDatePeriodList.begin();
         itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
      const DatePeriod* lDatePeriod_ptr = *itDatePeriod;
      assert (lDatePeriod_ptr != NULL);

      // Clone the current date-period
      DatePeriod& lCloneDatePeriod = cloneDatePeriod (*lDatePeriod_ptr);
      FacBomManager::addToListAndMap (lCloneAirportPair, lCloneDatePeriod);
      FacBomManager::linkWithParent (lCloneAirportPair, lCloneDatePeriod);
    }
  }

  return lCloneAirportPair;
}

// ////////////////////////////////////////////////////////////////////
DatePeriod& CmdCloneBomManager::
cloneDatePeriod (const DatePeriod& iDatePeriod) {

  DatePeriod& lCloneDatePeriod =
    FacCloneBom<DatePeriod>::instance().clone (iDatePeriod);

  // Check whether there are pos-channel objects
  const bool hasPosChannelList =
    BomManager::hasList<PosChannel> (iDatePeriod);
  if (hasPosChannelList == true) {
    // Browse the pos-channels
    const PosChannelList_T& lPosChannelList =
      BomManager::getList<PosChannel> (iDatePeriod);
    for (PosChannelList_T::const_iterator itPosChannel =
           lPosChannelList.begin();
         itPosChannel != lPosChannelList.end(); ++itPosChannel) {
      const PosChannel* lPosChannel_ptr = *itPosChannel;
      assert (lPosChannel_ptr != NULL);

      // Clone the current pos-channel
      PosChannel& lClonePosChannel = clonePosChannel (*lPosChannel_ptr);
      FacBomManager::addToListAndMap (lCloneDatePeriod, lClonePosChannel);
      FacBomManager::linkWithParent (lCloneDatePeriod, lClonePosChannel);
    }
  }

  return lCloneDatePeriod;
}


// ////////////////////////////////////////////////////////////////////
PosChannel& CmdCloneBomManager::
clonePosChannel (const PosChannel& iPosChannel) {
```

```
  PosChannel& lClonePosChannel =
    FacCloneBom<PosChannel>::instance().clone (iPosChannel);

  // Check whether there are time-period objects
  const bool hasTimePeriodList =
    BomManager::hasList<TimePeriod> (iPosChannel);
  if (hasTimePeriodList == true) {
    // Browse the time-periods
    const TimePeriodList_T& lTimePeriodList =
      BomManager::getList<TimePeriod> (iPosChannel);
    for (TimePeriodList_T::const_iterator itTimePeriod =
           lTimePeriodList.begin();
         itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
      const TimePeriod* lTimePeriod_ptr = *itTimePeriod;
      assert (lTimePeriod_ptr != NULL);

      // Clone the current time-period
      TimePeriod& lCloneTimePeriod = cloneTimePeriod (*lTimePeriod_ptr);
      FacBomManager::addToListAndMap (lClonePosChannel, lCloneTimePeriod);
      FacBomManager::linkWithParent (lClonePosChannel, lCloneTimePeriod);
    }
  }

  return lClonePosChannel;
}

// ////////////////////////////////////////////////////////////////////
TimePeriod& CmdCloneBomManager::
cloneTimePeriod (const TimePeriod& iTimePeriod) {

  TimePeriod& lCloneTimePeriod =
    FacCloneBom<TimePeriod>::instance().clone (iTimePeriod);

  // Check whether there are fare-feature objects
  const bool hasFareFeaturesList =
    BomManager::hasList<FareFeatures> (iTimePeriod);
  if (hasFareFeaturesList == true) {
    // Browse the fare-features
    const FareFeaturesList_T& lFareFeaturesList =
      BomManager::getList<FareFeatures> (iTimePeriod);
    for (FareFeaturesList_T::const_iterator itFF = lFareFeaturesList.begin();
         itFF != lFareFeaturesList.end(); ++itFF) {
      const FareFeatures* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Clone the current fare-feature
      FareFeatures& lCloneFareFeatures =
        cloneFeatures<FareFeatures> (*lFF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneFareFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneFareFeatures);
    }
  }

  // Check whether there are yield-feature objects
  const bool hasYieldFeaturesList =
    BomManager::hasList<YieldFeatures> (iTimePeriod);
  if (hasYieldFeaturesList == true) {
    // Browse the yield-features
    const YieldFeaturesList_T& lYieldFeaturesList =
      BomManager::getList<YieldFeatures> (iTimePeriod);
    for (YieldFeaturesList_T::const_iterator itYF =
           lYieldFeaturesList.begin();
         itYF != lYieldFeaturesList.end(); ++itYF) {
      const YieldFeatures* lYF_ptr = *itYF;
      assert (lYF_ptr != NULL);
```

```
      // Clone the current yield-feature
      YieldFeatures& lCloneYieldFeatures =
        cloneFeatures<YieldFeatures> (*lYF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneYieldFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneYieldFeatures);
    }
  }

  return lCloneTimePeriod;
}

// ////////////////////////////////////////////////////////////////////
template <typename FEATURE_TYPE>
FEATURE_TYPE& CmdCloneBomManager::
cloneFeatures (const FEATURE_TYPE& iFeatures) {

  FEATURE_TYPE& lCloneFeatures =
    FacCloneBom<FEATURE_TYPE>::instance().clone (iFeatures);

  // Check whether there are airline-class list objects
  const bool hasAirlineClassListList =
    BomManager::hasList<AirlineClassList> (iFeatures);
  if (hasAirlineClassListList == true) {
    // Browse the airline-class lists
    const AirlineClassListList_T& lAirlineClassList =
      BomManager::getList<AirlineClassList> (iFeatures);
    for (AirlineClassListList_T::const_iterator itACList =
           lAirlineClassList.begin();
         itACList != lAirlineClassList.end(); ++itACList) {
      const AirlineClassList* lACList_ptr = *itACList;
      assert (lACList_ptr != NULL);

      // Clone the current airline-class list
      AirlineClassList& lCloneAirlineClassList =
        cloneAirlineClassList (*lACList_ptr);
      FacBomManager::addToListAndMap (lCloneFeatures,
                                      lCloneAirlineClassList);
      FacBomManager::linkWithParent (lCloneFeatures,
                                     lCloneAirlineClassList);
    }
  }

  return lCloneFeatures;
}

// ////////////////////////////////////////////////////////////////////
AirlineClassList& CmdCloneBomManager::
cloneAirlineClassList (const AirlineClassList& iAirlineClassList) {

  AirlineClassList& lCloneAirlineClassList =
    FacCloneBom<AirlineClassList>::instance().clone (iAirlineClassList);

  return lCloneAirlineClassList;
}

// ////////////////////////////////////////////////////////////////////
FlightPeriod& CmdCloneBomManager::
cloneFlightPeriod (const FlightPeriod& iFlightPeriod) {

  FlightPeriod& lCloneFlightPeriod =
    FacCloneBom<FlightPeriod>::instance().clone (iFlightPeriod);

  // Check whether there are airline-class list objects
  const bool hasSegmentPeriodList =
    BomManager::hasList<SegmentPeriod> (iFlightPeriod);
  if (hasSegmentPeriodList == true) {
    // Browse the airline-class lists
```

```
      const SegmentPeriodList_T& lSegmentPeriodList =
        BomManager::getList<SegmentPeriod> (iFlightPeriod);
      for (SegmentPeriodList_T::const_iterator itSegmentPeriod =
             lSegmentPeriodList.begin();
           itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
        const SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
        assert (lSegmentPeriod_ptr != NULL);

        // Clone the current airline-class list
        SegmentPeriod& lCloneSegmentPeriod =
          cloneSegmentPeriod (*lSegmentPeriod_ptr);
        FacBomManager::addToListAndMap (lCloneFlightPeriod,
                                        lCloneSegmentPeriod);
        FacBomManager::linkWithParent (lCloneFlightPeriod,
                                       lCloneSegmentPeriod);
      }
    }

    return lCloneFlightPeriod;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod& CmdCloneBomManager::
  cloneSegmentPeriod (const SegmentPeriod& iSegmentPeriod) {

    SegmentPeriod& lCloneSegmentPeriod =
      FacCloneBom<SegmentPeriod>::instance().clone (iSegmentPeriod);

    return lCloneSegmentPeriod;
  }

}

/*!
```

6 C++ Class Storing the StdAir Context

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#if BOOST_VERSION >= 103900
#include <boost/make_shared.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/shared_ptr.hpp>
#endif // BOOST_VERSION >= 103900
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacCloneBom.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  STDAIR_ServiceContext::STDAIR_ServiceContext()
    : _cloneBomRoot (NULL),
      _persistentBomRoot (NULL),
      _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
    // Build the BomRoot object
```

```
  init();
}

// ////////////////////////////////////////////////////////////////////
STDAIR_ServiceContext::
STDAIR_ServiceContext (const STDAIR_ServiceContext& iServiceContext)
  : _cloneBomRoot (iServiceContext._cloneBomRoot),
    _persistentBomRoot (iServiceContext._persistentBomRoot),
    _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
  assert (false);
}

// ////////////////////////////////////////////////////////////////////
STDAIR_ServiceContext::~STDAIR_ServiceContext() {
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_ServiceContext::init() {
  //
  initBomRoot();
  initConfigHolder();
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_ServiceContext::initBomRoot() {
  _persistentBomRoot = &FacBom<BomRoot>::instance().create();
  initCloneBomRoot();
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_ServiceContext::initCloneBomRoot() {
  _cloneBomRoot =
    &FacCloneBom<BomRoot>::instance().clone(*_persistentBomRoot);
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_ServiceContext::initConfigHolder() {
  _configHolderPtr = boost::make_shared<ConfigHolderStruct> ();
}

// ////////////////////////////////////////////////////////////////////
const std::string STDAIR_ServiceContext::shortDisplay() const {
  std::ostringstream oStr;
  oStr << "STDAIR_ServiceContext -- " << _initType
       << " -- DB: " << _dbParams;
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string STDAIR_ServiceContext::display() const {
  std::ostringstream oStr;
  oStr << shortDisplay();
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string STDAIR_ServiceContext::describe() const {
  return shortDisplay();
}

// ////////////////////////////////////////////////////////////////////
BomRoot& STDAIR_ServiceContext::getPersistentBomRoot() const {
  assert (_persistentBomRoot != NULL);
  return *_persistentBomRoot;
}

// ////////////////////////////////////////////////////////////////////
```

```
  BomRoot& STDAIR_ServiceContext::getCloneBomRoot() const {
    assert (_cloneBomRoot != NULL);
    return *_cloneBomRoot;
  }

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct& STDAIR_ServiceContext::getConfigHolder() const {
    assert (_configHolderPtr != NULL);
    return *_configHolderPtr;
  }
}

/*!
```

7 People

7.1 Project Admins (and Developers)

- Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Anh Quan Nguyen <quannaus@users.sourceforge.net> (N)
- Gabrielle Sabatier <gsabatier@users.sourceforge.net> (N)

7.2 Retired Developers

- Mehdi Ayouni <mehdi.ayouni@gmail.com>
- Son Nguyen Kim <snguyenkim@users.sourceforge.net> (N)

7.3 Contributors

- Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

7.4 Distribution Maintainers

- Fedora/RedHat: Denis Arnaud <denis_arnaud@users.sourceforge.net> (N)
- Debian: Emmanuel Bastien <ebastien@users.sourceforge.net> (N)

Note:

(N) - Amadeus employees.

8 Coding Rules

In the following sections we describe the naming conventions which are used for files, classes, structures, local variables, and global variables.

8.1 Default Naming Rules for Variables

Variables names follow Java naming conventions. Examples:

- `lNumberOfPassengers`
- `lSeatAvailability`

8.2 Default Naming Rules for Functions

Function names follow Java naming conventions. Example:

- `int myFunctionName (const int& a, int b)`

8.3 Default Naming Rules for Classes and Structures

Each new word in a class or structure name should always start with a capital letter and the words should be separated with an under-score. Abbreviations are written with capital letters. Examples:

- `MyClassName`
- `MyStructName`

8.4 Default Naming Rules for Files

Files are named after the C++ class names.

Source files are named using `.cpp` suffix, whereas header files end with `.hpp` extension. Examples:

- `FlightDate.hpp`
- `SegmentDate.cpp`

8.5 Default Functionality of Classes

All classes that are configured by input parameters should include:

- default empty constructor
- one or more additional constructor(s) that takes input parameters and initializes the class instance
- setup function, preferably named `'setup'` or `'set_parameters'`

Explicit destructor functions are not required, unless they are needed. It shall not be possible to use any of the other member functions unless the class has been properly initiated with the input parameters.

9 Copyright and License

9.1 GNU LESSER GENERAL PUBLIC LICENSE

9.1.1 Version 2.1, February 1999


```
[This is the first released version of the Lesser GPL.  It also counts
as the successor of the GNU Library Public License, version 2, hence
the version number 2.1.]
```

9.2 Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to

permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

9.3 TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility

programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by

copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

9.3.1 NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

9.3.2 END OF TERMS AND CONDITIONS

9.4 How to Apply These Terms to Your New Programs

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
```

```
    Copyright (C) <year>  <name of author>

    This library is free software; you can redistribute it and/or
    modify it under the terms of the GNU Lesser General Public
    License as published by the Free Software Foundation; either
    version 2.1 of the License, or (at your option) any later version.

    This library is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
    Lesser General Public License for more details.

    You should have received a copy of the GNU Lesser General Public
    License along with this library; if not, write to the Free Software
    Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02110-1301  USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
  Yoyodyne, Inc., hereby disclaims all copyright interest in the
  library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

  <signature of Ty Coon>, 1 April 1990
  Ty Coon, President of Vice
```

That's all there is to it!

Source

10 Documentation Rules

10.1 General Rules

All classes in StdAir should be properly documented with Doxygen comments in include (`.hpp`) files. Source (`.cpp`) files should be documented according to a normal standard for well documented C++ code.

An example of how the interface of a class shall be documented in StdAir is shown here:

```
/*!
 * \brief Brief description of MyClass here
 *
 * Detailed description of MyClass here. With example code if needed.
 */
class MyClass {
public:
  //! Default constructor
  MyClass(void) { setup_done = false; }

  /*!
   * \brief Constructor that initializes the class with parameters
   *
   * Detailed description of the constructor here if needed
   *
   * \param[in] param1 Description of \a param1 here
   * \param[in] param2 Description of \a param2 here
   */
  MyClass(TYPE1 param1, TYPE2 param2) { setup(param1, param2); }

  /*!
```

```
   * \brief Setup function for MyClass
   *
   * Detailed description of the setup function here if needed
   *
   * \param[in] param1 Description of \a param1 here
   * \param[in] param2 Description of \a param2 here
   */
  void setup(TYPE1 param1, TYPE2 param2);

  /*!
   * \brief Brief description of memberFunction1
   *
   * Detailed description of memberFunction1 here if needed
   *
   * \param[in]     param1 Description of \a param1 here
   * \param[in]     param2 Description of \a param2 here
   * \param[in,out] param3 Description of \a param3 here
   * \return Description of the return value here
   */
  TYPE4 memberFunction1(TYPE1 param1, TYPE2 param2, TYPE3 &param3);

private:

  bool _setupDone;         /*!< Variable that checks if the class is properly
                                initialized with parameters */
  TYPE1 _privateVariable1; //!< Short description of _privateVariable1 here
  TYPE2 _privateVariable2; //!< Short description of _privateVariable2 here
};
```

10.2 File Header

All files should start with the following header, which include Doxygen's `\file`, `\brief` and `\author` tags, `$Date$` and `$Revisions$` CVS tags, and a common copyright note:

```
/*!
 * \file
 * \brief Brief description of the file here
 * \author Names of the authors who contributed to this code
 * \date Date
 *
 * Detailed description of the file here if needed.
 *
 * ---------------------------------------------------------------------------
 *
 * StdAir - C++ Standard Airline IT Object Library
 *
 * Copyright (C) 2009-2010  (\see authors file for a list of contributors)
 *
 * \see copyright file for license information
 *
 * ---------------------------------------------------------------------------
 */
```

10.3 Grouping Various Parts

All functions must be added to a Doxygen group in order to appear in the documentation. The following code example defines the group `'my_group'`:

```
/*!
 * \defgroup my_group Brief description of the group here
 *
 * Detailed description of the group here
```

```
*/
```

The following example shows how to document the function `myFunction` and how to add it to the group `my_group`:

```
/*!
 * \brief Brief description of myFunction here
 * \ingroup my_group
 *
 * Detailed description of myFunction here
 *
 * \param[in] param1 Description of \a param1 here
 * \param[in] param2 Description of \a param2 here
 * \return Description of the return value here
 */
TYPE3 myFunction(TYPE1 param1, TYPE2 &param2);
```

11 Main features

A short list of the main features of StdAir is given below sorted in different categories. Many more features and functions exist and for these we refer to the reference documentation.

11.1 Standard Airline IT Business Object Model (BOM)

- (Airline) Network-related classes:
  - Network, ReachableUniverse
- (Air) Travel-related classes:
  - TravelSolution, OriginDestination,
- (Airline) Inventory-related classes:
  - Inventory, FlightDate, SegmentDate, SegmentCabin, BookingClass, LegDate, LegCabin, Bucket
- (Airline) Schedule-related classes:
  - FlightPeriod, SegmentPeriod, LegPeriod
- (Simulated) Passenger-related demand classes:
  - DemandStream, BookingRequest
- (Air) Price-related classes:
  - YieldStore

11.2 Architecture of the StdAir library

- Separate structure and content classes
- `Boost.Fusion`

12 Make a Difference

Do not ask what StdAir can do for you. Ask what you can do for StdAir.

You can help us to develop the StdAir library. There are always a lot of things you can do:

- Start using StdAir
- Tell your friends about StdAir and help them to get started using it
- If you find a bug, report it to us (on the dedicated Sourceforge's Trac Web site). Without your help we can never hope to produce a bug free code.
- Help us to improve the documentation by providing information about documentation bugs
- Answer support requests in the StdAir discussion forums on SourceForge. If you know the answer to a question, help others to overcome their StdAir problems.
- Help us to improve our algorithms. If you know of a better way (e.g., that is faster or requires less memory) to implement some of our algorithms, then let us know.
- Help to port StdAir to new platforms. If you manage to compile StdAir on a new platform, then tell how you did it.
- Send your code. If you have a good StdAir compatible code, which you can release under the LGPL, and you think it should be included in StdAir, then send it to the communauty.
- Become an StdAir developer. Send us (see the People page) an e-mail and tell what you can do for StdAir.

13 Make a new release

13.1 Introduction

This document describes briefly the recommended procedure of releasing a new version of StdAir using a Linux development machine and the SourceForge project site.

The following steps are required to make a release of the distribution package.

13.2 Initialisation

Clone locally the full Git project:

```
cd ~
mkdir -p dev/sim
cd ~/dev/sim
git clone git://stdair.git.sourceforge.net/gitroot/stdair/stdair stdairgit
cd stdairgit
git checkout trunk
```

13.3 Branch creation

Create the branch, on your local clone, corresponding to the new release (say, 0.5.0):

```
cd ~/dev/sim/stdairgit
git checkout trunk
git checkout -b 0.5.0
```

Update the version in the various build system files, replacing 99.99.99 by the correct version number:

```
vi CMakeLists.txt
vi autogen.sh
```

Update the version and add a change-log in the ChangeLog and in the RPM specification files:

```
vi ChangeLog
vi stdair.spec
```

13.4 Commit and publish the release branch

Commit the new release:

```
cd ~/dev/sim/stdairgit
git add -A
git commit -m "[Release 0.5.0] Release of version 0.5.0."
git push
```

13.5 Update the change-log in the trunk as well

Update the change-log in the ChangeLog and RPM specification files:

```
cd ~/dev/sim/stdairgit
git checkout trunk
vi ChangeLog
vi stdair.spec
```

Commit the change-logs and publish the trunk (main development branch):

```
git commit -m "[Doc] Integrated the change-log of the release 0.5.0."
git push
```

13.6 Create distribution packages

Create the distribution packages using the following command:

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.5.0 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make check && make dist
```

This will configure, compile and check the package. The output packages will be named, for instance, `stdair-0.5.0.tar.gz` and `stdair-0.5.0.tar.bz2`.

13.7 Generation the RPM packages

Optionally, generate the RPM package (for instance, for `Fedora`/`RedHat`):

```
cd ~/dev/sim/stdairgit
git checkout 0.5.0
rm -rf build && mkdir -p build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-99.99.99 \
 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
make dist
```

To perform this step, rpm-build, rpmlint and rpmdevtools have to be available on the system.

```
cp stdair.spec ~/dev/packages/SPECS \
  && cp stdair-0.5.0.tar.bz2 ~/dev/packages/SOURCES
cd ~/dev/packages/SPECS
rpmbuild -ba stdair.spec
rpmlint -i ../SPECS/stdair.spec ../SRPMS/stdair-0.5.0-1.fc15.src.rpm \
  ../RPMS/noarch/stdair-* ../RPMS/i686/stdair-*
```

13.8 Update distributed change log

Update the `NEWS` and `ChangeLog` files with appropriate information, including what has changed since the previous release. Then commit and push the changes into the `StdAir's Git repository`.

13.9 Create the binary package, including the documentation

Create the binary package, which includes HTML and PDF documentation, using the following command:

```
make package
```

The output binary package will be named, for instance, `stdair-0.5.0-Linux.tar.bz2`. That package contains both the HTML and PDF documentation. The binary package contains also the executables and shared libraries, as well as C++ header files, but all of those do not interest us for now.

13.10 Upload the files to SourceForge

Upload the distribution and documentation packages to the SourceForge server. Check `SourceForge help page on uploading software`.

13.11 Upload the documentation to SourceForge

In order to update the Web site files, either:

- `synchronise them with rsync and SSH`:

  ```
  cd ~/dev/sim/stdairgit
  git checkout 0.5.0
  rsync -aiv doc/html/ doc/latex/refman.pdf joe,stdair@web.sourceforge.net:htdocs/
  ```

  where `-aiv` options mean:

  - `-a`: archive/mirror mode; equals `-rlptgoD` (no `-H`, `-A`, `-X`)
  - `-v`: increase verbosity
  - `-i`: output a change-summary for all updates

  - Note the trailing slashes (/) at the end of both the source and target directories. It means that the content of the source directory (`doc/html`), rather than the directory itself, has to be copied into the content of the target directory.

- or use the `SourceForge Shell service`.

13.12 Make a new post

- submit a new entry in the `SourceForge project-related news feed`
- make a new post on the `SourceForge hosted WordPress blog`
- and update, if necessary, `Trac tickets`.

13.13 Send an email on the announcement mailing-list

Finally, you should send an announcement to `stdair-announce@lists.sourceforge.net` (see `https://lists.sourceforge.net/lists/listinfo/stdair-announce` for the archives)

14 Installation

14.1 Table of Contents

14.2 Fedora/RedHat Linux distributions

Note that on `Fedora`/`RedHat` Linux distributions, RPM packages are available and can be installed with your usual package manager. For instance:

```
yum -y install stdair-devel stdair-doc
```

RPM packages can also be available on the `SourceForge download site`.

14.3 StdAir Requirements

StdAir should compile without errors or warnings on most GNU/Linux systems, on UNIX systems like Solaris SunOS, and on POSIX based environments for Microsoft Windows like Cygwin or MinGW with MSYS. It can be also built on Microsoft Windows NT/2000/XP/Vista/7 using Microsoft's Visual C++ .NET, but our support for this compiler is limited. For GNU/Linux, SunOS, Cygwin and MinGW we assume that you have at least the following GNU software installed on your computer:

- GNU Autotools:
  - autoconf,
  - automake,
  - libtool,
  - make, version 3.72.1 or later (check version with 'make --version')
- GCC - GNU C++ Compiler (g++), version 4.3.x or later (check version with 'gcc --version')
- Boost - C++ STL extensions, version 1.35 or later (check version with 'grep "define BOOST_-LIB_VERSION" /usr/include/boost/version.hpp')
- MySQL - Database client libraries, version 5.0 or later (check version with 'mysql --version')
- SOCI - C++ database client library wrapper, version 3.0.0 or later (check version with 'soci-config --version')

Optionally, you might need a few additional programs: Doxygen, LaTeX, Dvips and Ghostscript, to generate the HTML and PDF documentation.

We strongly recommend that you use recent stable releases of the GCC, if possible. We do not actively work on supporting older versions of the GCC, and they may therefore (without prior notice) become unsupported in future releases of StdAir.

14.4 Basic Installation

Briefly, the shell commands './cmake .. && make install' should configure, build and install this package. The following more-detailed instructions are generic; see the 'README' file for instructions specific to this package. Some packages provide this 'INSTALL' file but do not implement all of the features documented below. The lack of an optional feature in a given package is not necessarily a bug. More recommendations for GNU packages can be found in the info page corresponding to "Makefile Conventions: (standards)Makefile Conventions".

The 'cmake' shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a 'Makefile' in each directory of the package. It may also create one or more '.h' files containing system-dependent definitions. Finally, it creates a 'CMakeCache.txt' cache file that you can refer to in the future to recreate the current configuration, and files 'CMakeFiles' containing compiler output (useful mainly for debugging 'cmake').

It can also use an optional file (typically called 'config.cache' and enabled with '--cache-file=config.cache' or simply '-C') that saves the results of its tests to speed up reconfiguring. Caching is disabled by default to prevent problems with accidental use of stale cache files.

If you need to do unusual things to compile the package, please try to figure out how 'configure' could check whether to do them, and mail diffs or instructions to the address given in the 'README' so they can be considered for the next release. If you are using the cache, and at some point 'config.cache' contains results you don't want to keep, you may remove or edit it.

The file 'CMakeLists.txt' is used to create the 'Makefile' files.

The simplest way to compile this package is:

1. 'cd' to the directory containing the package's source code and type './cmake ..' to configure the package for your system. Running 'cmake' is generally fast. While running, it prints some messages telling which features it is checking for.
2. Type 'make' to compile the package.
3. Optionally, type 'make check'to run any self-tests that come with the package, generally using the just-built uninstalled binaries.
4. Type 'make install' to install the programs and any data files and documentation. When installing into a prefix owned by root, it is recommended that the package be configured and built as a regular user, and only the 'make install' phase executed with root privileges.
5. You can remove the program binaries and object files from the source code directory by typing 'make clean'. To also remove the files that 'configure' created (so you can compile the package for a different kind of computer), type 'make distclean'. There is also a 'make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.
6. Often, you can also type 'make uninstall' to remove the installed files again. In practice, not all packages have tested that uninstallation works correctly, even though it is required by the GNU Coding Standards.

14.5 Compilers and Options

Some systems require unusual options for compilation or linking that the 'cmake' script does not know about. Run './cmake --help' for details on some of the pertinent environment variables.

You can give 'cmake' initial values for configuration parameters by setting variables in the command line or in the environment. Here is an example:

```
./cmake CC=c99 CFLAGS=-g LIBS=-lposix
```

See also:

Defining Variables for more details.

14.6 Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you can use GNU 'make'. 'cd' to the directory where you want the object files and executables to go and

run the 'configure' script. 'configure' automatically checks for the source code in the directory that 'configure' is in and in '..'. This is known as a "VPATH" build.

With a non-GNU 'make', it is safer to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use 'make distclean' before reconfiguring for another architecture.

On MacOS X 10.5 and later systems, you can create libraries and executables that work on multiple system types--known as "fat" or "universal" binaries--by specifying multiple '-arch' options to the compiler but only a single '-arch' option to the preprocessor. Like this:

```
./configure CC="gcc -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
           CXX="g++ -arch i386 -arch x86_64 -arch ppc -arch ppc64" \
           CPP="gcc -E" CXXCPP="g++ -E"
```

This is not guaranteed to produce working output in all cases, you may have to build one architecture at a time and combine the results using the 'lipo' tool if you have problems.

14.7 Installation Names

By default, 'make install' installs the package's commands under '/usr/local/bin', include files under '/usr/local/include', etc. You can specify an installation prefix other than '/usr/local' by giving 'configure' the option '--prefix=PREFIX', where PREFIX must be an absolute file name.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you pass the option '--exec-prefix=PREFIX' to 'configure', the package uses PREFIX as the prefix for installing programs and libraries. Documentation and other data files still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like '--bindir=DIR' to specify different values for particular kinds of files. Run 'configure --help' for a list of the directories you can set and what kinds of files go in them. In general, the default for these options is expressed in terms of '${prefix}', so that specifying just '--prefix' will affect all of the other directory specifications that were not explicitly provided.

The most portable way to affect installation locations is to pass the correct locations to 'configure'; however, many packages provide one or both of the following shortcuts of passing variable assignments to the 'make install' command line to change installation locations without having to reconfigure or recompile.

The first method involves providing an override variable for each affected directory. For example, 'make install prefix=/alternate/directory' will choose an alternate location for all directory configuration variables that were expressed in terms of '${prefix}'. Any directories that were specified during 'configure',

but not in terms of '${prefix}', must each be overridden at install time for the entire installation to be relocated. The approach of makefile variable overrides for each directory variable is required by the GNU Coding Standards, and ideally causes no recompilation. However, some platforms have known limitations with the semantics of shared libraries that end up requiring recompilation when using this method, particularly noticeable in packages that use GNU Libtool.

The second method involves providing the 'DESTDIR' variable. For example, 'make install DESTDIR=/alternate/directory' will prepend '/alternate/directory' before all installation names. The approach of 'DESTDIR' overrides is not required by the GNU Coding Standards, and does not work on platforms that have drive letters. On the other hand, it does better at avoiding recompilation issues, and works well even when some directory options were not specified in terms of '${prefix}' at 'configure' time.

14.8 Optional Features

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving 'cmake' the option '--program-prefix=PREFIX' or '--program-suffix=SUFFIX'.

Some packages pay attention to '--enable-FEATURE' options to 'configure', where FEATURE indicates an optional part of the package. They may also pay attention to '--with-PACKAGE' options, where PACKAGE is something like 'gnu-as' or 'x' (for the X Window System). The 'README' should mention any '--enable-' and '--with-' options that the package recognizes.

For packages that use the X Window System, 'configure' can usually find the X include and library files automatically, but if it doesn't, you can use the 'configure' options '--x-includes=DIR' and '--x-libraries=DIR' to specify their locations.

Some packages offer the ability to configure how verbose the execution of 'make' will be. For these packages, running './configure --enable-silent-rules' sets the default to minimal output, which can be overridden with 'make V=1'; while running './configure --disable-silent-rules' sets the default to verbose, which can be overridden with 'make V=0'.

14.9 Particular systems

On HP-UX, the default C compiler is not ANSI C compatible. If GNU CC is not installed, it is recommended to use the following options in order to use an ANSI C compiler:

```
./configure CC="cc -Ae -D_XOPEN_SOURCE=500"
```

and if that doesn't work, install pre-built binaries of GCC for HP-UX.

On OSF/1 a.k.a. Tru64, some versions of the default C compiler cannot parse its '<wchar.h>' header file. The option '-nodtk' can be used as

a workaround. If GNU CC is not installed, it is therefore recommended to try

```
./configure CC="cc"
```

and if that doesn't work, try

```
./configure CC="cc -nodtk"
```

On Solaris, don't put '/usr/ucb' early in your 'PATH'. This directory contains several dysfunctional programs; working variants of these programs are available in '/usr/bin'. So, if you need '/usr/ucb' in your 'PATH', put it _after_ '/usr/bin'.

On Haiku, software installed for all users goes in '/boot/common', not '/usr/local'. It is recommended to use the following options:

```
./cmake -DCMAKE_INSTALL_PREFIX=/boot/common
```

14.10 Specifying the System Type

There may be some features 'configure' cannot figure out automatically, but needs to determine by the type of machine the package will run on. Usually, assuming the package is built to be run on the _same_ architectures, 'configure' can figure that out, but if it prints a message saying it cannot guess the machine type, give it the '--build=TYPE' option. TYPE can either be a short name for the system type, such as 'sun4', or a canonical name which has the form CPU-COMPANY-SYSTEM

where SYSTEM can have one of these forms:

- OS
- KERNEL-OS

See the file 'config.sub' for the possible values of each field. If 'config.sub' isn't included in this package, then this package doesn't need to know the machine type.

If you are _building_ compiler tools for cross-compiling, you should use the option '--target=TYPE' to select the type of system they will produce code for.

If you want to _use_ a cross compiler, that generates code for a platform different from the build platform, you should specify the "host" platform (i.e., that on which the generated programs will eventually be run) with '--host=TYPE'.

14.11 Sharing Defaults

If you want to set default values for 'configure' scripts to share, you can create a site shell script called 'config.site' that gives

default values for variables like 'CC', 'cache_file', and 'prefix'. 'configure' looks for 'PREFIX/share/config.site' if it exists, then 'PREFIX/etc/config.site' if it exists. Or, you can set the 'CONFIG_-SITE' environment variable to the location of the site script. A warning: not all 'configure' scripts look for a site script.

14.12 Defining Variables

Variables not defined in a site shell script can be set in the environment passed to 'configure'. However, some packages may run configure again during the build, and the customized values of these variables may be lost. In order to avoid this problem, you should set them in the 'configure' command line, using 'VAR=value'. For example:

```
./configure CC=/usr/local2/bin/gcc
```

causes the specified 'gcc' to be used as the C compiler (unless it is overridden in the site shell script).

Unfortunately, this technique does not work for 'CONFIG_SHELL' due to an Autoconf bug. Until the bug is fixed you can use this workaround:

```
CONFIG_SHELL=/bin/bash /bin/bash ./configure CONFIG_SHELL=/bin/bash
```

14.13 'cmake' Invocation

'cmake' recognizes the following options to control how it operates.

- '--help', '-h' print a summary of all of the options to 'configure', and exit.
- '--help=short', '--help=recursive' print a summary of the options unique to this package's 'configure', and exit. The 'short' variant lists options used only in the top level, while the 'recursive' variant lists options also present in any nested packages.
- '--version', '-V' print the version of Autoconf used to generate the 'configure' script, and exit.
- '--cache-file=FILE' enable the cache: use and save the results of the tests in FILE, traditionally 'config.cache'. FILE defaults to '/dev/null' to disable caching.
- '--config-cache', '-C' alias for '--cache-file=config.cache'.
- '--quiet', '--silent', '-q' do not print messages saying which checks are being made. To suppress all normal output, redirect it to '/dev/null' (any error messages will still be shown).
- '--srcdir=DIR' look for the package's source code in directory DIR. Usually 'configure' can determine that directory automatically.
- '--prefix=DIR' use DIR as the installation prefix.

See also:

Installation Names for more details, including other options available for fine-tuning the installation locations.

- '--no-create', '-n' run the configure checks, but stop before creating any output files.

'cmake' also accepts some other, not widely useful, options. Run 'cmake --help' for more details.

The 'cmake' script produces an ouput like this:

```
cmake -DCMAKE_INSTALL_PREFIX=/home/user/dev/deliveries/stdair-0.50.0 \
 -DLIB_SUFFIX=64 -DCMAKE_BUILD_TYPE:STRING=Debug -DINSTALL_DOC:BOOL=ON ..
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/lib64/ccache/gcc
-- Check for working C compiler: /usr/lib64/ccache/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/lib64/ccache/c++
-- Check for working CXX compiler: /usr/lib64/ccache/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Requires Git without specifying any version
-- Current Git revision name: e8beb4d11ff9b1af6b3f3e9ff1e92250aee0291a trunk
-- Requires PythonLibs-2.7
-- Found PythonLibs: /usr/lib64/libpython2.7.so (Required is at least version "2.7")
-- Found PythonLibs 2.7
-- Requires Boost-1.41
-- Boost version: 1.46.0
-- Found the following Boost libraries:
--   program_options
--   date_time
--   iostreams
--   serialization
--   filesystem
--   unit_test_framework
--   python
-- Found Boost version: 1.46.0
-- Found BoostWrapper: /usr/include (Required is at least version "1.41")
-- Requires ZeroMQ-2.0
-- Found ZeroMQ: /usr/lib64/libzmq.so (Required is at least version "2.0")
-- Found ZeroMQ version: 2.1
-- Requires MySQL-5.1
-- Using mysql-config: /usr/bin/mysql_config
-- Found MySQL: /usr/lib64/mysql/libmysqlclient.so (Required is at least version "5.1")
-- Found MySQL version: 5.5.14
-- Requires SOCI-3.0
-- Using soci-config: /usr/bin/soci-config
-- SOCI headers are buried
-- Found SOCI: /usr/lib64/libsoci_core.so (Required is at least version "3.0")
-- Found SOCIMySQL: /usr/lib64/libsoci_mysql.so (Required is at least version "3.0")
-- Found SOCI with MySQL back-end support version: 3.0.0
-- Requires Doxygen-1.7
-- Found Doxygen: /usr/bin/doxygen
-- Found DoxygenWrapper: /usr/bin/doxygen (Required is at least version "1.7")
-- Found Doxygen version: 1.7.4
-- Had to set the linker language for 'stdairlib' to CXX
-- Had to set the linker language for 'stdairuicllib' to CXX
-- Test 'StdAirTest' to be built with 'MPBomRoot.cpp;MPInventory.cpp;StandardAirlineITTestSuite.cpp'
--
-- ==============================================================
-- ----------------------------------
-- ---     Project Information    ---
```

```
-- -----------------------------------
-- PROJECT_NAME ................... : stdair
-- PACKAGE_PRETTY_NAME ............ : StdAir
-- PACKAGE ........................ : stdair
-- PACKAGE_NAME ................... : STDAIR
-- PACKAGE_VERSION ................ : 0.50.0
-- GENERIC_LIB_VERSION ............ : 0.50.0
-- GENERIC_LIB_SOVERSION .......... : 99.99
--
-- -----------------------------------
-- ---     Build Configuration    ---
-- -----------------------------------
-- Modules to build ............... : stdair
-- Libraries to build/install ..... : stdairlib;stdairuicllib
-- Binaries to build/install ...... : stdair
-- Modules to test ................ : stdair
-- Binaries to test ............... : StdAirTesttst
--
-- * Module ....................... : stdair
--   + Layers to build ............ : .;basic;bom;factory;dbadaptor;command;service
--   + Dependencies on other layers :
--   + Libraries to build/install . : stdairlib;stdairuicllib
--   + Executables to build/install : stdair
--   + Tests to perform ........... : StdAirTesttst
--
-- BUILD_SHARED_LIBS .............. : ON
-- CMAKE_BUILD_TYPE ............... : Debug
--  * CMAKE_C_FLAGS ............... :
--  * CMAKE_CXX_FLAGS ............. : -O2 -g -pipe -Wall -Wp,-D_FORTIFY_SOURCE=2 -fexceptions -fstack-prot
--  * BUILD_FLAGS ................. :
--  * COMPILE_FLAGS ............... :
-- CMAKE_MODULE_PATH .............. : /home/user/dev/sim/stdair/stdairgithub/config/
-- CMAKE_INSTALL_PREFIX ........... : /home/user/dev/deliveries/stdair-0.50.0
--
-- * Doxygen:
--   - DOXYGEN_VERSION ............ : 1.7.4
--   - DOXYGEN_EXECUTABLE ......... : /usr/bin/doxygen
--   - DOXYGEN_DOT_EXECUTABLE ..... : /usr/bin/dot
--   - DOXYGEN_DOT_PATH ........... : /usr/bin
--
-- -----------------------------------
-- --- Installation Configuration ---
-- -----------------------------------
-- INSTALL_LIB_DIR ................ : /home/user/dev/deliveries/stdair-0.50.0/lib64
-- INSTALL_BIN_DIR ................ : /home/user/dev/deliveries/stdair-0.50.0/bin
-- INSTALL_INCLUDE_DIR ............ : /home/user/dev/deliveries/stdair-0.50.0/include
-- INSTALL_DATA_DIR ............... : /home/user/dev/deliveries/stdair-0.50.0/share
-- INSTALL_SAMPLE_DIR ............. : /home/user/dev/deliveries/stdair-0.50.0/share/stdair/samples
-- INSTALL_DOC .................... : ON
--
-- -----------------------------------
-- ---   Packaging Configuration  ---
-- -----------------------------------
-- CPACK_PACKAGE_CONTACT .......... : Denis Arnaud <denis_arnaud - at - users dot sourceforge dot net>
-- CPACK_PACKAGE_VENDOR ........... : Denis Arnaud
-- CPACK_PACKAGE_VERSION .......... : 0.50.0
-- CPACK_PACKAGE_DESCRIPTION_FILE . : /home/user/dev/sim/stdair/stdairgithub/README
-- CPACK_RESOURCE_FILE_LICENSE .... : /home/user/dev/sim/stdair/stdairgithub/COPYING
-- CPACK_GENERATOR ................ : TBZ2
-- CPACK_DEBIAN_PACKAGE_DEPENDS ... :
-- CPACK_SOURCE_GENERATOR ......... : TBZ2;TGZ
-- CPACK_SOURCE_PACKAGE_FILE_NAME . : stdair-0.50.0
--
-- ----------------------------------
-- ---     External libraries    ---
-- ----------------------------------
--
```

```
-- * Python:
--   - PYTHONLIBS_VERSION ......... : 2.7
--   - PYTHON_LIBRARIES ........... : /usr/lib64/libpython2.7.so
--   - PYTHON_INCLUDE_PATH ........ : /usr/include/python2.7
--   - PYTHON_INCLUDE_DIRS ........ : /usr/include/python2.7
--   - PYTHON_DEBUG_LIBRARIES ..... :
--   - Python_ADDITIONAL_VERSIONS . :
--
-- * ZeroMQ:
--   - ZeroMQ_VERSION ............. : 2.1
--   - ZeroMQ_LIBRARIES ........... : /usr/lib64/libzmq.so
--   - ZeroMQ_INCLUDE_DIR ......... : /usr/include
--
-- * Boost:
--   - Boost_VERSION .............. : 104600
--   - Boost_LIB_VERSION .......... : 1_46
--   - Boost_HUMAN_VERSION ........ : 1.46.0
--   - Boost_INCLUDE_DIRS ......... : /usr/include
--   - Boost required components .. : program_options;date_time;iostreams;serialization;filesystem;unit_te
--   - Boost required libraries ... : optimized;/usr/lib64/libboost_iostreams-mt.so;debug;/usr/lib64/libbo
--
-- * MySQL:
--   - MYSQL_VERSION .............. : 5.5.14
--   - MYSQL_INCLUDE_DIR .......... : /usr/include/mysql
--   - MYSQL_LIBRARIES ............ : /usr/lib64/mysql/libmysqlclient.so
--
-- * SOCI:
--   - SOCI_VERSION ............... : 3.0.0
--   - SOCI_INCLUDE_DIR ........... : /usr/include/soci
--   - SOCIMYSQL_INCLUDE_DIR ...... : /usr/include/soci
--   - SOCI_LIBRARIES ............. : /usr/lib64/libsoci_core.so
--   - SOCIMYSQL_LIBRARIES ........ : /usr/lib64/libsoci_mysql.so
--
-- Change a value with: cmake -D<Variable>=<Value>
-- ==============================================================
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/user/dev/sim/stdair/stdairgithub/build
```

It is recommended that you check if your library has been compiled and linked properly and works as expected. To do so, you should execute the testing process 'make check'. As a result, you should obtain a similar report:

```
[  0%] Built target hdr_cfg_stdair
[ 97%] Built target stdairlib
[100%] Built target StdAirTesttst
Scanning dependencies of target check_stdairtst
Test project /home/user/dev/sim/stdair/stdairgithub/build/test/stdair
    Start 1: StdAirTesttst
1/1 Test #1: StdAirTesttst ...................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 1

Total Test time (real) =   0.27 sec
[100%] Built target check_stdairtst
Scanning dependencies of target check
[100%] Built target check
```

Check if all the executed tests PASSED. If not, please contact us by filling a bug-report.

Finally, you should install the compiled and linked library, include files and (optionally) HTML and PDF documentation by typing:

```
make install
```

Depending on the PREFIX settings during configuration, you might need the root (administrator) access to perform this step.

Eventually, you might invoke the following command

```
make clean
```

to remove all files created during compilation process, or even

```
cd ~/dev/sim/stdairgit
rm -rf build && mkdir build
cd build
```

to remove everything.

15 Linking with StdAir

15.1 Table of Contents

15.2 Introduction

There are two convenient methods of linking your programs with the StdAir library. The first one employs the `'pkg-config'` command (see http://pkgconfig.freedesktop.org/), whereas the second one uses `'stdair-config'` script. These methods are shortly described below.

15.3 Using the pkg-config command

`'pkg-config'` is a helper tool used when compiling applications and libraries. It helps you insert the correct compiler and linker options. The syntax of the `'pkg-config'` is as follows:

```
pkg-config <options> <library_name>
```

For instance, assuming that you need to compile an StdAir based program `'my_prog.cpp'`, you should use the following command:

```
g++ `pkg-config --cflags stdair` -o my_prog my_prog.cpp \
 `pkg-config --libs stdair`
```

For more information see the `'pkg-config'` man pages.

15.4 Using the stdair-config script

StdAir provides a shell script called `'stdair-config'`, which is installed by default in `'$prefix/bin'` (`'/usr/local/bin'`) directory. It can be used to simplify compilation and linking of StdAir based programs. The usage of this script is quite similar to the usage of the `'pkg-config'` command.

Assuming that you need to compile the program `'my_prog`.cpp' you can now do that with the following command:

```
g++ 'stdair-config --cflags' -o my_prog my_prog.cpp 'stdair-config --libs'
```

A list of `'stdair-config'` options can be obtained by typing:

```
stdair-config --help
```

If the `'stdair-config'` command is not found by your shell, you should add its location `'$prefix/bin'` to the `PATH` environment variable, e.g.:

```
export PATH=/usr/local/bin:$PATH
```

15.5 M4 macro for the GNU Autotools

A M4 macro file is delivered with StdAir, namely 'stdair.m4', which can be found in, e.g., '/usr/share/aclocal'. When used by a 'configure' script, thanks to he `'AM_PATH_STDAIR'` macro (specified in the M4 macro file), the following Makefile variables are then defined:

- `'STDAIR_VERSION'` (e.g., defined to 0.2.0)
- `'STDAIR_CFLAGS'` (e.g., defined to `'-I${prefix}/include'`)
- `'STDAIR_LIBS'` (e.g., defined to `'-L${prefix}/lib -lstdair'`)

15.6 Using StdAir with dynamic linking

When using static linking some of the library routines in StdAir are copied into your executable program. This can lead to unnecessary large executables. To avoid having too large executable files you may use dynamic linking instead. Dynamic linking means that the actual linking is performed when the program is executed. This requires that the system is able to locate the shared StdAir library file during your program execution. If you install the StdAir library using a non-standard prefix, the `'LD_LIBRARY_PATH'` environment variable might be used to inform the linker of the dynamic library location, e.g.:

```
export LD_LIBRARY_PATH=<StdAir installation prefix>/lib:$LD_LIBRARY_PATH
```

16 Test Rules

This section describes how the functionality of the StdAir library should be verified. In the `'test/stdair'` subdirectory, test source files are provided. All functionality should be tested using these test source files.

16.1 The Test Source Files

Each new StdAir module/class should be accompanied with a test source file. The test source file is an implementation in C++ that tests the functionality of a function/class or a group of functions/classes called test suites. The test source file should test relevant parameter settings and input/output relations to guarantee correct functionality of the corresponding classes/functions. The test source files should be maintained using version control and updated whenever new functionality is added to the StdAir library.

The test source file should print relevant data to a standard output that can be used to verify the functionality. All relevant parameter settings should be tested.

The test source file should be placed in the `'test/stdair'` subdirectory and should have a name ending with `'TestSuite.cpp'`.

16.2 The Reference File

Consider a test source file named `'YieldTestSuite.cpp'`. A reference file named `'YieldTestSuite.ref'` should accompany the test source file. The reference file contains a reference printout of the standard output generated when running the test program. The reference file should be maintained using version control and updated according to the test source file.

16.3 Testing StdAir Library

One can compile and execute all test programs from the `'test/stdair'` sub-directory by typing:

```
% make check
```

after successful compilation of the StdAir library.

17 Users Guide

17.1 Table of Contents

17.2 Introduction

The `StdAir` library contains classes for airline business management. This document does not cover all the aspects of the `StdAir` library. It does however explain the most important things you need to know in order to start using `StdAir`.

17.3 Get Started

17.3.1 Get the StdAir library

17.3.2 Build the StdAir project

To run the configuration script the first time, go to the top directory (where the StdAir package has been un-packed), and issue either of the following two commands, depending on whether the StdAir project has been checked out from the Subversion repository or downloaded as a tar-ball package from the Sourceforge Web site:

- `./autogen.sh`
- `./configure`

17.3.3 Build and Run the Tests

17.3.4 Install the StdAir Project (Binaries, Documentation)

17.4 Exploring the Predefined BOM Tree

`StdAir` predefines a BOM (Business Object Model) tree specific to the airline IT arena.

17.4.1 Airline Distribution BOM Tree

- `stdair::TravelSolutionStruct`

17.4.2 Airline Network BOM Tree

- `stdair::FlightPeriod`

17.4.3 Airline Inventory BOM Tree

- `stdair::Inventory`
- `stdair::FlightDate`

17.4.3.1 Airline Inventory Marketing BOM Tree

- `stdair::SegmentDate`
- `stdair::SegmentCabin`
- `stdair::FareFamily`
- `stdair::BookingClass`

17.4.3.2 Airline Inventory Operating BOM Tree

- `stdair::LegDate`
- `stdair::LegCabin`
- `stdair::Bucket`

17.5 Extending the BOM Tree

18 Supported Systems

18.1 Table of Contents

18.2 Introduction

This page is intended to provide a list of StdAir supported systems, i.e. the systems on which configuration, installation and testing process of the StdAir library has been sucessful. Results are grouped based on minor release number. Therefore, only the latest tests for bug-fix releases are included. Besides, the information on this page is divided into sections dependent on the operating system.

Where necessary, some extra information is given for each tested configuration, e.g. external libraries installed, configuration commands used, etc.

If you manage to compile, install and test the StdAir library on a system not mentioned below, please let us know, so we could update this database.

18.3 StdAir 3.10.x

18.3.1 Linux Systems

18.3.1.1 Fedora Core 4 with ATLAS

- **Platform**: Intel Pentium 4
- **Operating System**: Fedora Core 4 (x86)
- **Compiler**: g++ (GCC) 4.0.2 20051125
- **StdAir release**: 3.10.0
- **External Libraries**: From FC4 distribution:
  - `fftw3.i386-3.0.1-3`
  - `fftw3-devel.i386-3.0.1-3`
  - `atlas-sse2.i386-3.6.0-8.fc4`
  - `atlas-sse2-devel.i386-3.6.0-8.fc4`
  - `blas.i386-3.0-35.fc4`
  - `lapack.i386-3.0-35.fc4`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% CXXFLAGS="-O3 -pipe -march=pentium4" ./configure
```

- **Date**: March 7, 2006
- **Tester**: Tony Ottosson

18.3.1.2 Gentoo Linux with ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler(s)**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/acml-3.0.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ACML
% eselect lapack set ACML
```

StdAir configured with:

```
% export CPPFLAGS="-I/usr/include/acml"
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.1.3 Gentoo Linux with ATLAS

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-atlas-3.6.0-r1`
  - `sci-libs/lapack-atlas-3.6.0`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% eselect blas set ATLAS
% eselect lapack set ATLAS
```

StdAir configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.1.4 Gentoo Linux with MKL

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86 arch)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED

- **Comments**: StdAir configured using the following commands:

```
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/32"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: February 28, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.1.5 Gentoo Linux with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium M Centrino
- **Operating System**: Gentoo Linux 2006.0 (x86)
- **Compiler**: g++ (GCC) 3.4.5
- **StdAir release**: 3.10.1
- **External Libraries**: Compiled and installed from portage tree:
  - `sci-libs/fftw-3.1`
  - `sci-libs/blas-reference-19940131-r2`
  - `sci-libs/cblas-reference-20030223`
  - `sci-libs/lapack-reference-3.0-r2`
- **Tests Status**: All tests PASSED
- **Comments**: BLAS and LAPACK libs set by using the following system commands:

```
% blas-config reference
% lapack-config reference
```

  StdAir configured with:

```
% ./configure --with-blas="-lblas"
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.1.6 Red Hat Enterprise Linux with StdAir External

- **Platform**: Intel Pentium 4
- **Operating System**: Red Hat Enterprise Linux AS release 4 (Nahant Update 2)
- **Compiler**: g++ (GCC) 3.4.4 20050721 (Red Hat 3.4.4-2)
- **StdAir release**: 3.10.0
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package
- **Tests Status**: All tests PASSED
- **Date**: March 7, 2006
- **Tester**: Erik G. Larsson

18.3.1.7 SUSE Linux 10.0 with NetLib's BLAS and LAPACK

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **StdAir release**: 3.10.0
- **External Libraries**: BLAS, LAPACK and FFTW libraries installed from OpenSuse 10.0 RPM repository:
  - `blas-3.0-926`
  - `lapack-3.0-926`
  - `fftw3-3.0.1-114`
  - `fftw3-threads-3.0.1-114`
  - `fftw3-devel-3.0.1-114`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% ./configure --with-lapack="/usr/lib64/liblapack.so.3"
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.1.8 SUSE Linux 10.0 with MKL

- **Platform**: Intel Pentium 4 CPU 3.20GHz (64-bit)
- **Operating System**: SUSE Linux 10.0 (x86_64)
- **Compiler(s)**: g++ (GCC) 4.0.2
- **StdAir release**: 3.10.0
- **External Libraries**: Intel Math Kernel Library (MKL) 8.0.1 installed manually in the following directory: `/opt/intel/mkl/8.0.1`
- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export CXXFLAGS="-m64 -march=nocona -O3 -pipe"
% export LDFLAGS="-L/opt/intel/mkl/8.0.1/lib/em64t"
% export CPPFLAGS="-I/opt/intel/mkl/8.0.1/include"
% ./configure
```

- **Date**: March 1, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2 Windows Systems

18.3.2.1 Microsoft Windows XP with Cygwin

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **StdAir release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`
  - `lapack-3.0-4`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2.2 Microsoft Windows XP with Cygwin and ATLAS

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **StdAir release**: 3.10.1
- **External Libraries**: Installed from Cygwin's repository:
  - `fftw-3.0.1-2`
  - `fftw-dev-3.0.1-1`

  ATLAS BLAS and LAPACK libraries from StdAir External 2.1.1 package configured using:

```
% ./configure --enable-atlas --disable-fftw
```

- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% ./configure
```

- **Date**: March 31, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2.3 Microsoft Windows XP with Cygwin and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, Cygwin 1.5.19-4
- **Compiler(s)**: g++ (GCC) 3.4.4 (cygming special)
- **StdAir release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/cygdrive/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2.4 Microsoft Windows XP with MinGW, MSYS and ACML

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **StdAir release**: 3.10.2
- **External Libraries**: ACML version 3.1.0 (`acml3.1.0-32-win32-g77.exe`) installed into a default directory, i.e. `"c:\Program Files\AMD\acml3.1.0"`
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/c/Progra~1/AMD/acml3.1.0/gnu32/lib"
% export CPPFLAGS="-I/c/Progra~1/AMD/acml3.1.0/gnu32/include"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2.5 Microsoft Windows XP with MinGW, MSYS and StdAir External

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2, MinGW 5.0.2, MSYS 1.0.10
- **Compiler(s)**: g++ (GCC) 3.4.4 (mingw special)
- **StdAir release**: 3.10.5
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.2.0 package
- **Tests Status**: All tests PASSED
- **Comments**: Only static library can be built. StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-Wall -O3 -march=athlon-tbird -pipe"
% ./configure --disable-html-doc
```

- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.2.6 Microsoft Windows XP with MS Visual C++ and Intel MKL

- **Platform**: AMD Sempron 3000+
- **Operating System**: Microsoft Windows XP SP2
- **Compiler(s)**: Microsoft Visual C++ 2005 .NET
- **StdAir release**: 3.10.5
- **External Libraries**: Intel Math Kernel Library (MKL) 8.1 installed manually in the following directory: `"C:\Program Files\Intel\MKL\8.1"`
- **Tests Status**: Not fully tested. Some StdAir based programs compiled and run with success.
- **Comments**: Only static library can be built. StdAir built by opening the `"win32\stdair.vcproj"` project file in MSVC++ and executing `"Build -> Build Solution"` command from menu.
- **Date**: August 11, 2006
- **Tester**: Adam Piatyszek (ediap)

18.3.3 Unix Systems

18.3.3.1 SunOS 5.9 with StdAir External

- **Platform**: SUNW, Sun-Blade-100 (SPARC)
- **Operating System**: SunOS 5.9 Generic_112233-10
- **Compiler(s)**: g++ (GCC) 3.4.5

- **StdAir release**: 3.10.2
- **External Libraries**: BLAS, CBLAS, LAPACK and FFTW libraries from StdAir External 2.1.1 package. The following configuration command has been used:

```
% export CFLAGS="-mcpu=ultrasparc -O2 -pipe -funroll-all-loops"
% ./configure
```

- **Tests Status**: All tests PASSED
- **Comments**: StdAir configured with:

```
% export LDFLAGS="-L/usr/local/lib"
% export CPPFLAGS="-I/usr/local/include"
% export CXXFLAGS="-mcpu=ultrasparc -O2 -pipe"
% ./configure --enable-debug
```

- **Date**: May 15, 2006
- **Tester**: Adam Piatyszek (ediap)

19 StdAir Supported Systems (Previous Releases)

19.1 StdAir 3.9.1

19.2 StdAir 3.9.0

19.3 StdAir 3.8.1

20 Tutorials

20.1 Table of Contents

20.2 Introduction

This page contains some tutorial examples that will help you getting started using StdAir. Most examples show how to construct some simple business objects, i.e., instances of the so-named Business Object Model (BOM).

20.2.1 Preparing the StdAir Project for Development

The source code for these examples can be found in the `batches` and `test/stdair` directories. They are compiled along with the rest of the `StdAir` project. See the User Guide (Users Guide) for more details on how to build the `StdAir` project.

20.3 Build a Predefined BOM Tree

A few steps:

- Instanciate the BOM Root Object
- Instanciate the (Airline) Inventory Object
- Link the Inventory Object with the BOM Root

20.3.1 Instanciate the BOM Root Object

First, a BOM root object (i.e., a root for all the classes in the project) is instantiated by the `stdair::STDAIR_ServiceContext` context object, when the `stdair::STDAIR_Service` is itself instantiated. The corresponding `StdAir` type (class) is `stdair::BomRoot`.

In the following sample, that object is named `ioBomRoot`, and is given as input/output parameter of the `stdair::CmdBomManager::buildSampleBom()` method:

```
\textcolor{keywordtype}{void} CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) \{
```

20.3.2 Instanciate the (Airline) Inventory Object

An airline inventory object can then be instantiated. Let us give it the "BA" airline code (corresponding to `British Airways`) as the object key. That is, an object (let us name it `lBAKey`) of type (class) `stdair::InventoryKey` has first to be instantiated.

```
\textcolor{keyword}{const} InventoryKey lBAKey (lAirlineCodeBA);
```

Thanks to that key, an airline inventory object, i.e. of type (class) `stdair::Inventory`, can be instantiated. Let us name that airline inventory object `lBAInv`.

```
Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
```

20.3.3 Link the Inventory Object with the BOM Root

Then, both objects have to be linked: the airline inventory object (`stdair::Inventory`) has to be linked with the root of the BOM tree (`stdair::BomRoot`). That operation is as simple as using the `stdair::FacBomManager::addToListAndMap()` method:

```
FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
FacBomManager::linkWithParent (ioBomRoot, lBAInv);
```

20.3.4 Build Another Airline Inventory

Another airline inventory object, corresponding to the Air France (`Air France`) company, is instantiated the same way:

```
\textcolor{keyword}{const} InventoryKey lAFKey (lAirlineCodeAF);
Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
FacBomManager::linkWithParent (ioBomRoot, lAFInv);
```

See the corresponding full program (C++ Class Building Sample StdAir BOM Trees) for more details.

20.3.5 Dump The BOM Tree Content

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
\textcolor{keyword}{const} InventoryList\_T& lInventoryList =
  BomManager::getList<Inventory> (iBomRoot);
```

... and browsed:

```
\textcolor{keywordflow}{for} (InventoryList\_T::const\_iterator itInv = lInventoryList.begin();
     itInv != lInventoryList.end(); ++itInv, ++invIdx) \{
  \textcolor{keyword}{const} Inventory* lInv\_ptr = *itInv;
  assert (lInv\_ptr != NULL);

  \textcolor{comment}{// Retrieve the inventory key (airline code)}
  \textcolor{keyword}{const} \hyperlink{a00509_ac0ab9f765621de3140cc27d6472f5581}{AirlineCode_T}& lAir

  \textcolor{comment}{// Display only the requested inventories}
  \textcolor{keywordflow}{if} (iAirlineCode == \textcolor{stringliteral}{"all"} || iAirlineCode == lAi
    \textcolor{comment}{// Get the list of flight-dates for that inventory}
    list (oStream, *lInv\_ptr, invIdx, iFlightNumber);
  \}
\}
\}

\textcolor{comment}{// ////////////////////////////////////////////////////////////////////}
\textcolor{keywordtype}{void} BomDisplay::list (std::ostream& oStream, \textcolor{keyword}{const} Invent
                       \textcolor{keyword}{const} \textcolor{keywordtype}{unsigned} \textcolor{keywordty
                       \textcolor{keyword}{const} \hyperlink{a00509_abb24a0796f29fe921b6dbe3ba0fa638e}{F
  \textcolor{comment}{// Save the formatting flags for the given STL output stream}
  FlagSaver flagSaver (oStream);

  \textcolor{comment}{// Check whether there are FlightDate objects}
```

```
  \textcolor{keywordflow}{if} (BomManager::hasMap<FlightDate> (iInventory) == \textcolor{keyword}{false}
    \textcolor{keywordflow}{return};
  \}

  \textcolor{comment}{//}
  \textcolor{keyword}{const} \hyperlink{a00509_ac0ab9f765621de3140cc27d6472f5581}{AirlineCode_T}& lAirli
  oStream << iInventoryIndex << \textcolor{stringliteral}{". "} << lAirlineCode << std::endl;

  \textcolor{comment}{// Browse the flight-dates}
  \textcolor{keywordtype}{unsigned} \textcolor{keywordtype}{short} lCurrentFlightNumber = 0;
  \textcolor{keywordtype}{unsigned} \textcolor{keywordtype}{short} flightNumberIdx = 0;
  \textcolor{keywordtype}{unsigned} \textcolor{keywordtype}{short} departureDateIdx = 1;
  \textcolor{keyword}{const} \hyperlink{a00509_a161cab8007596da006451289628a1137}{FlightDateMap_T}& lFli
    BomManager::getMap<FlightDate> (iInventory);
  \textcolor{keywordflow}{for} (FlightDateMap\_T::const\_iterator itFD = lFlightDateList.begin();
       itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) \{
    \textcolor{keyword}{const} FlightDate* lFD\_ptr = itFD->second;
    assert (lFD\_ptr != NULL);

    \textcolor{comment}{// Retrieve the key of the flight-date}
    \textcolor{keyword}{const} \hyperlink{a00509_abb24a0796f29fe921b6dbe3ba0fa638e}{FlightNumber_T}& lFl
    \textcolor{keyword}{const} \hyperlink{a00509_a049eb52315b4e9bb21a411f8b9d9012e}{Date_T}& lFlightDate

    \textcolor{comment}{// Display only the requested flight number}
    \textcolor{keywordflow}{if} (iFlightNumber == 0 || iFlightNumber == lFlightNumber) \{
      \textcolor{comment}{//}
      \textcolor{keywordflow}{if} (lCurrentFlightNumber != lFlightNumber) \{
        lCurrentFlightNumber = lFlightNumber;
        ++flightNumberIdx; departureDateIdx = 1;
        oStream << \textcolor{stringliteral}{"  "} << iInventoryIndex << \textcolor{stringliteral}{"."}
                << lAirlineCode << lFlightNumber << std::endl;
      \}

      oStream << \textcolor{stringliteral}{"    "} << iInventoryIndex << \textcolor{stringliteral}{"."}
              << \textcolor{stringliteral}{"."} << departureDateIdx << \textcolor{stringliteral}{". "}
              << lAirlineCode << lFlightNumber << \textcolor{stringliteral}{" / "} << lFlightDateDate
              << std::endl;
    \}
  \}
\}

\textcolor{comment}{// ////////////////////////////////////////////////////////////////////}
\textcolor{keywordtype}{void} BomDisplay::listAirportPairDateRange (std::ostream& oStream,
                                        \textcolor{keyword}{const} BomRoot& iBomRoot) \{
  \textcolor{comment}{// Save the formatting flags for the given STL output stream}
  FlagSaver flagSaver (oStream);

  \textcolor{comment}{// Check whether there are AirportPair objects}
  \textcolor{keywordflow}{if} (BomManager::hasList<AirportPair> (iBomRoot) == \textcolor{keyword}{false}
    \textcolor{keywordflow}{return};
  \}

  \textcolor{keyword}{const} \hyperlink{a00509_a3910e03bffd66a6d8bef3d20c98bd0eb}{AirportPairList_T}& lA
    BomManager::getList<AirportPair> (iBomRoot);
  \textcolor{keywordflow}{for} (AirportPairList\_T::const\_iterator itAir = lAirportPairList.begin();
       itAir != lAirportPairList.end(); ++itAir ) \{
    \textcolor{keyword}{const} AirportPair* lAir\_ptr = *itAir;
    assert (lAir\_ptr != NULL);

    \textcolor{comment}{// Check whether there are date-period objects}
    assert (BomManager::hasList<DatePeriod> (*lAir\_ptr) == \textcolor{keyword}{true});

    \textcolor{comment}{// Browse the date-period objects}
    \textcolor{keyword}{const} \hyperlink{a00509_a0cb61b0f5151e050d83c5574f0f55cd0}{DatePeriodList_T}& l
      BomManager::getList<DatePeriod> (*lAir\_ptr);

    \textcolor{keywordflow}{for} (DatePeriodList\_T::const\_iterator itDP = lDatePeriodList.begin();
```

```
          itDP != lDatePeriodList.end(); ++itDP) \{
       \textcolor{keyword}{const} DatePeriod* lDP\_ptr = *itDP;
       assert (lDP\_ptr != NULL);

       \textcolor{comment}{// Display the date-period object}
       oStream << lAir\_ptr->describeKey()
               <<\textcolor{stringliteral}{" / "} << lDP\_ptr->describeKey() << std::endl;
     \}

   \}
 \}

 \textcolor{comment}{// ////////////////////////////////////////////////////////////////////}
 \textcolor{keywordtype}{void} BomDisplay::csvDisplay (std::ostream& oStream,
```

See the corresponding full program (C++ Utility Class Browsing and Dumping the StdAir BOM Tree) for more details.

20.3.6 Result of the Tutorial Program

When the `stdair.cpp` program is run (with the `-b` option), the output should look like:

```
[D]../../batches/stdair.cpp:243: Welcome to stdair
[D]../../../stdair/command/CmdBomManager.cpp:41: StdAir will build the BOM tree f
      rom built-in specifications.
[D]../../batches/stdair.cpp:286:
================================================================
BomRoot:  -- ROOT --
================================================================
++++++++++++++++++++++++++++++++++++++++++++++++
Inventory: BA
++++++++++++++++++++++++++++++++++++++++++++++++
****************************************
FlightDate: BA9, 2011-Jun-10
****************************************
****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
BA9 2011-Jun-10, LHR-BKK, 2011-Jun-10, 21:45:00, 2011-Jun-11, 15:40:00, 11:05:00,
       1, 06:50:00, 9900, 0,
BA9 2011-Jun-10, BKK-SYD, 2011-Jun-11, 17:05:00, 2011-Jun-12, 15:40:00, 09:05:00,
       1, 13:30:00, 8100, 0,
****************************************
****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
BA9 2011-Jun-10, LHR-BKK 2011-Jun-10, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0
      , 3.52965e-319, 0, 0,
BA9 2011-Jun-10, BKK-SYD 2011-Jun-11, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0, 0
      , 0, 0, 0,
****************************************
****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
****************************************
****************************************
SegmentCabins:
--------------
```

```
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
BA9 2011-Jun-10, LHR-SYD 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
BA9 2011-Jun-10, LHR-BKK 2011-Jun-10, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
BA9 2011-Jun-10, BKK-SYD 2011-Jun-11, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
*****************************************
*****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
BA9 2011-Jun-10, LHR-SYD 2011-Jun-10, Y, EcoSaver, Q, 0 (0), 0, 0, 0, 0, 0 (0), 0
      , 0, 0, 0, 0, 0,
++++++++++++++++++++++++++++++++++++++++++++++++
Inventory: AF
++++++++++++++++++++++++++++++++++++++++++++++++
*****************************************
FlightDate: AF84, 2011-Mar-20
*****************************************
*****************************************
Leg-Dates:
----------
Flight, Leg, BoardDate, BoardTime, OffDate, OffTime, Date Offset, Time Offset, El
      apsed, Distance, Capacity,
AF84 2011-Mar-20, CDG-SFO, 2011-Mar-20, 10:40:00, 2011-Mar-20, 12:50:00, 11:10:00
      , 0, -09:00:00, 9900, 0,
*****************************************
*****************************************
LegCabins:
----------
Flight, Leg, Cabin, OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, Comm
      Space, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 9, 9, 0,
      0, 0, 0, 0,
*****************************************
*****************************************
Buckets:
--------
Flight, Leg, Cabin, Yield, AU/SI, SS, AV,
*****************************************
*****************************************
SegmentCabins:
--------------
Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, CommSpace, AvPool, BP,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, EcoSaver, 0, 0, 0, 0, 9, 0,
*****************************************
*****************************************
Subclasses:
-----------
Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), Nego, NS%, OB%, Bkgs, GrpBks
       (pdg), StfBkgs, WLBkgs, ETB, ClassAvl, RevAvl, SegAvl,
AF84 2011-Mar-20, CDG-SFO 2011-Mar-20, Y, EcoSaver, Q, 0 (0), 0, 0, 0, 0, 0 (0),
      0, 0, 0, 0, 0, 0,
```

See the corresponding full program (Command-Line Utility to Demonstrate Typical StdAir Usage) for more details.

20.4 Extend the Pre-Defined BOM Tree

Now that we master how to instantiate the pre-defined StdAir classes, let us see how to extend that BOM.

20.4.1 Extend an Airline Inventory Object

For instance, let us assume that some (IT) provider (e.g., you) would like to have a specific implementation of the `Inventory` object. The corresponding class has just to extend the `stdair::Inventory` class:

```
\textcolor{keyword}{namespace }myprovider \{
  \textcolor{keyword}{class }Inventory : \textcolor{keyword}{public} stdair::Inventory \{
```

The STL containers have to be defined accordingly too:

```
  \textcolor{keyword}{typedef} std::list<Inventory*> InventoryList\_T;
```

See the full class definition (Specific Implementation of an Airline Inventory) and implementation (Specific Implementation of an Airline Inventory) for more details.

20.4.2 Build the Specific BOM Objects

The BOM root object (`stdair::BomRoot`) is instantiated the classical way:

```
  \textcolor{keyword}{const} std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) can be instantiated the same way as a standard Inventory (`stdair::Inventory`) would be:

```
  \textcolor{keyword}{const} \hyperlink{a00088}{stdair::InventoryKey} lBAKey (lBAAirlineCode);
  myprovider::Inventory& lBAInv =
    \hyperlink{a00064}{stdair::FacBom<myprovider::Inventory>::instance}().create (lBAKey);
```

Then, the specific implementation of the airline inventory object (`myprovider::Inventory`) is linked to the root of the BOM tree (`stdair::BomRoot`) the same way as the standard Inventory (`stdair::Inventory`) would be:

```
  \hyperlink{a00065_a15bd081544ad82cbe13dfebf10600662}{stdair::FacBomManager::addToList} (lBomRoot, lBAInv
```

Another specific airline inventory object is instantiated the same way:

```
  \textcolor{keyword}{const} \hyperlink{a00088}{stdair::InventoryKey} lAFKey (lAFAirlineCode);
  myprovider::Inventory& lAFInv =
    \hyperlink{a00064}{stdair::FacBom<myprovider::Inventory>::instance}().create (lAFKey);
  \hyperlink{a00065_a15bd081544ad82cbe13dfebf10600662}{stdair::FacBomManager::addToList} (lBomRoot, lAFInv
```

From the `BomRoot` (of type `stdair::BomRoot`) object instance, the list of specific airline inventories (of type `stdair::Inventory`) can then be retrieved...

```
  \textcolor{keyword}{const} \hyperlink{a00509_aa2443c204cfcb7ce31e90308b9a63d71}{myprovider::InventoryLis
      stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
```

... and browsed:

```
\textcolor{keywordflow}{for} (myprovider::InventoryList\_T::const\_iterator itInv =
       lInventoryList.begin(); itInv != lInventoryList.end();
     ++itInv, ++idx) \{
  \textcolor{keyword}{const} myprovider::Inventory* lInv\_ptr = *itInv;
  BOOST\_REQUIRE (lInv\_ptr != NULL);

  BOOST\_CHECK\_EQUAL (lInventoryKeyArray[idx], lInv\_ptr->describeKey());
  BOOST\_CHECK\_MESSAGE (lInventoryKeyArray[idx] == lInv\_ptr->describeKey(),
                        \textcolor{stringliteral}{"They inventory key, '"} << lInventoryKeyArray[idx]
                        << \textcolor{stringliteral}{"', does not match that of the Inventory object: '"}
                        << lInv\_ptr->describeKey() << \textcolor{stringliteral}{"'"});
\}
```

20.4.3 Result of the Tutorial Program

When this program is run, the output should look like:

```
Inventory: BA
Inventory: AF
```

See the corresponding full program (Command-Line Test to Demonstrate How To Extend StdAir BOM) for more details.

21 Command-Line Utility to Demonstrate Typical StdAir Usage

```
 */
// STL
#include <cassert>
#include <iostream>
#include <sstream>
#include <fstream>
#include <string>
// Boost (Extended STL)
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/program_options.hpp>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
// StdAir
#include <stdair/stdair_types.hpp>
#include <stdair/bom/BomArchive.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/config/stdair-paths.hpp>

// //////// Constants //////
const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");

const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
                                                   "/schedule01.csv");

const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOL = false;

const bool K_STDAIR_DEFAULT_BUILT_FOR_CRS = false;
```

```
const int K_STDAIR_EARLY_RETURN_STATUS = 99;

// ///////// Parsing of Options & Configuration /////////
// A helper function to simplify the main part.
template<class T> std::ostream& operator<< (std::ostream& os,
                                            const std::vector<T>& v) {
  std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
  return os;
}

int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
                       bool& ioIsForRMOL, bool& ioIsForCRS,
                       stdair::Filename_T& ioInputFilename,
                       std::string& ioLogFilename) {
  // Default for the built-in input
  ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;

  // Default for the RMOL input
  ioIsForRMOL = K_STDAIR_DEFAULT_BUILT_FOR_RMOL;

  // Default for the CRS input
  ioIsForCRS = K_STDAIR_DEFAULT_BUILT_FOR_CRS;

  // Declare a group of options that will be allowed only on command line
  boost::program_options::options_description generic ("Generic options");
  generic.add_options()
    ("prefix", "print installation prefix")
    ("version,v", "print version string")
    ("help,h", "produce help message");

  // Declare a group of options that will be allowed both on command
  // line and in config file

  boost::program_options::options_description config ("Configuration");
  config.add_options()
    ("builtin,b",
     "The sample BOM tree can be either built-in or parsed from an input file. Th
      at latter must then be given with the -i/--input option")
    ("rmol,r",
     "Build a sample BOM tree for RMOL (i.e., a dummy flight-date with a single l
      eg-cabin)")
    ("crs,c",
     "Build a sample BOM tree for CRS")
    ("input,i",
     boost::program_options::value< std::string >(&ioInputFilename)->default_valu
      e(K_STDAIR_DEFAULT_INPUT_FILENAME),
     "(CVS) input file for the demand distributions")
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(
      K_STDAIR_DEFAULT_LOG_FILENAME),
     "Filename for the logs")
    ;

  // Hidden options, will be allowed both on command line and
  // in config file, but will not be shown to the user.
  boost::program_options::options_description hidden ("Hidden options");
  hidden.add_options()
    ("copyright",
     boost::program_options::value< std::vector<std::string> >(),
     "Show the copyright (license)");

  boost::program_options::options_description cmdline_options;
  cmdline_options.add(generic).add(config).add(hidden);

  boost::program_options::options_description config_file_options;
  config_file_options.add(config).add(hidden);
```

```
boost::program_options::options_description visible ("Allowed options");
visible.add(generic).add(config);

boost::program_options::positional_options_description p;
p.add ("copyright", -1);

boost::program_options::variables_map vm;
boost::program_options::
  store (boost::program_options::command_line_parser (argc, argv).
         options (cmdline_options).positional(p).run(), vm);

std::ifstream ifs ("stdair.cfg");
boost::program_options::store (parse_config_file (ifs, config_file_options),
                               vm);
boost::program_options::notify (vm);

if (vm.count ("help")) {
  std::cout << visible << std::endl;
  return K_STDAIR_EARLY_RETURN_STATUS;
}

if (vm.count ("version")) {
  std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
  return K_STDAIR_EARLY_RETURN_STATUS;
}

if (vm.count ("prefix")) {
  std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
  return K_STDAIR_EARLY_RETURN_STATUS;
}

if (vm.count ("builtin")) {
  ioIsBuiltin = true;
}

if (vm.count ("rmol")) {
  ioIsForRMOL = true;

  // The RMOL sample tree takes precedence over the default built-in BOM tree
  ioIsBuiltin = false;
}

if (vm.count ("crs")) {
  ioIsForCRS = true;

  // The RMOL sample tree takes precedence over the default built-in BOM tree
  ioIsBuiltin = false;
}

const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
          << std::endl;

const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
          << std::endl;

if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
  if (vm.count ("input")) {
    ioInputFilename = vm["input"].as< std::string >();
    std::cout << "Input filename is: " << ioInputFilename << std::endl;

  } else {
    std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
```

```
                << "or -i/--input options must be specified" << std::endl;
    }
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}


// ///////////////// M A I N ////////////////////
int main (int argc, char* argv[]) {

  // State whether the BOM tree should be built-in or parsed from an
  // input file
  bool isBuiltin;

  // State whether a sample BOM tree should be built for RMOL.
  bool isForRMOL;

  // State whether a sample BOM tree should be built for the CRS.
  bool isForCRS;

  // Input file name
  stdair::Filename_T lInputFilename;

  // Output log File
  std::string lLogFilename;

  // Call the command-line option parser
  const int lOptionParserStatus =
    readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
                       lInputFilename, lLogFilename);

  if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
    return 0;
  }

  // Set the log parameters
  std::ofstream logOutputFile;
  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // DEBUG
  STDAIR_LOG_DEBUG ("Welcome to stdair");

  // Check wether or not a (CSV) input file should be read
  if (isBuiltin == true || isForRMOL == true || isForCRS == true) {

    if (isForRMOL == true) {
      // Build the sample BOM tree for RMOL
      stdairService.buildDummyInventory (300);

    } else if (isForCRS == true) {
      //
      stdair::TravelSolutionList_T lTravelSolutionList;
      stdairService.buildSampleTravelSolutions (lTravelSolutionList);

      // Build the sample BOM tree for CRS
      const stdair::BookingRequestStruct& lBookingRequest =
```

```
        stdairService.buildSampleBookingRequest();

      // DEBUG: Display the travel solution and booking request
      STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.display());

      const std::string& lCSVDump =
        stdairService.csvDisplay (lTravelSolutionList);
      STDAIR_LOG_DEBUG (lCSVDump);

    } else {
      assert (isBuiltin == true);

      // Build a sample BOM tree
      stdairService.buildSampleBom();
    }

  } else {
    // Read the input file
    //stdairService.readFromInputFile (lInputFilename);

    // DEBUG
    STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
                      << " and build the corresponding BOM tree.");
  }

  // DEBUG: Display the whole persistent BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Close the Log outputFile
  logOutputFile.close();

  /*
    Note: as that program is not intended to be run on a server in
    production, it is better not to catch the exceptions. When it
    happens (that an exception is throwned), that way we get the
    call stack.
  */

  return 0;
}

/*!
```

22 Specific Implementation of a BOM Root

```
 */
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir Test
#include <test/stdair/MPBomRoot.hpp>

namespace myprovider {

  // ////////////////////////////////////////////////////////////////////
  BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot::~BomRoot () {
  }
```

```
}
/*!
```

23 Specific Implementation of a BOM Root

```
 */

// ///////////////////////////////////////////////////////////////////////
// Import section
// ///////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/bom/BomRoot.hpp>

namespace myprovider {

  class BomRoot : public stdair::BomRoot {
  public:
    // /////////// Display support methods /////////
    std::string toString() const { return describeKey(); }

    const std::string describeKey() const { return std::string (""); }

  public:
    BomRoot (const Key_T&);
    ~BomRoot();
    BomRoot ();
    BomRoot (const BomRoot&);
  };

}
/*!
```

24 Specific Implementation of an Airline Inventory

```
 */
// ///////////////////////////////////////////////////////////////////////
// Import section
// ///////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
// StdAir Test
#include <test/stdair/MPInventory.hpp>

namespace myprovider {

  // ///////////////////////////////////////////////////////////////////////
  Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
  }

  // ///////////////////////////////////////////////////////////////////////
  Inventory::~Inventory () {
  }

  // ///////////////////////////////////////////////////////////////////////
  std::string Inventory::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
```

```
  }

  // ///////////////////////////////////////////////////////////////////
  const std::string Inventory::describeKey() const {
    return _key.toString();
  }

}
/*!
```

25 Specific Implementation of an Airline Inventory

```
 */

// ///////////////////////////////////////////////////////////////////
// Import section
// ///////////////////////////////////////////////////////////////////
// STL
#include <list>
// StdAir
#include <stdair/bom/Inventory.hpp>

namespace myprovider {

  class Inventory : public stdair::Inventory {
  public:
    // /////////// Display support methods /////////
    std::string toString() const;

    const std::string describeKey() const;

  public:
    Inventory (const Key_T&);
    ~Inventory();
    Inventory ();
    Inventory (const Inventory&);
  };

  // ////////// Type definitions /////////
  typedef std::list<Inventory*> InventoryList_T;

}
/*!
```

26 Command-Line Test to Demonstrate How To Extend StdAir BOM

```
 */
// ///////////////////////////////////////////////////////////////////
// Import section
// ///////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <fstream>
#include <string>
// Boost MPL
#include <boost/mpl/push_back.hpp>
#include <boost/mpl/vector.hpp>
#include <boost/mpl/at.hpp>
#include <boost/mpl/assert.hpp>
#include <boost/type_traits/is_same.hpp>
```

```
// Boost Unit Test Framework (UTF)
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MAIN
#define BOOST_TEST_MODULE StdAirTest
#if BOOST_VERSION >= 103900
#include <boost/test/unit_test.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/test/test_tools.hpp>
#include <boost/test/results_reporter.hpp>
#include <boost/test/unit_test_suite.hpp>
#include <boost/test/output_test_stream.hpp>
#include <boost/test/unit_test_log.hpp>
#include <boost/test/framework.hpp>
#include <boost/test/detail/unit_test_parameters.hpp>
#endif // BOOST_VERSION >= 103900
// Boost Serialisation
#include <boost/archive/text_oarchive.hpp>
#include <boost/archive/text_iarchive.hpp>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/STDAIR_Service.hpp>
#include <stdair/basic/float_utils.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacBomManager.hpp>
// StdAir Test Suite
#include <test/stdair/StdairTestLib.hpp>
#include <test/stdair/MPInventory.hpp>

namespace boost_utf = boost::unit_test;

#if BOOST_VERSION >= 103900

// (Boost) Unit Test XML Report
std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");

struct UnitTestConfig {
  UnitTestConfig() {
    boost_utf::unit_test_log.set_stream (utfReportStream);
    boost_utf::unit_test_log.set_format (boost_utf::XML);
    boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
    // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_te
      sts);
  }

  ~UnitTestConfig() {
  }
};


// /////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (float_comparison_test) {
  float a = 0.2f;
  a = 5*a;
  const float b = 1.0f;
```

```
  // Test the Boost way
  BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
  BOOST_CHECK_CLOSE (a, b, 0.0001);

  // Test the Google way
  const FloatingPoint<float> lhs (a), rhs (b);
  BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),
                       "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
}

BOOST_AUTO_TEST_CASE (mpl_structure_test) {
  const stdair::ClassCode_T lBookingClassCodeA ("A");
  const stdair_test::BookingClass lA (lBookingClassCodeA);
  const stdair_test::Cabin lCabin (lA);

  BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
  BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
                       "The cabin key, '" << lCabin.toString()
                       << "' is not equal to '" << lBookingClassCodeA << "'");

  // MPL
  typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
  typedef boost::mpl::push_back<MPL_BookingClass,
                                stdair_test::Cabin>::type types;

  if (boost::is_same<stdair_test::BookingClass,
                     stdair_test::Cabin::child>::value == false) {
    BOOST_ERROR ("The two types mut be equal, but are not");
  }

  if (boost::is_same<boost::mpl::at_c<types, 1>::type,
                     stdair_test::Cabin>::value == false) {
    BOOST_ERROR ("The type must be stdair_test::Cabin, but is not");
  }
}

BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // Retrieve (a reference on) the top of the persistent BOM tree
  stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
```

```
                      << "', should be equal to '" << lBomRootString
                      << "', but is not.");

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // DEBUG: Display the whole BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
  // Step 0.0: initialisation
  // Create the root of a Bom tree (i.e., a BomRoot object)
  stdair::BomRoot& lBomRoot =
    stdair::FacBom<stdair::BomRoot>::instance().create();

  // Step 0.1: Inventory level
  // Create an Inventory (BA)
  const stdair::AirlineCode_T lBAAirlineCode ("BA");
  const stdair::InventoryKey lBAKey (lBAAirlineCode);
  myprovider::Inventory& lBAInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
  stdair::FacBomManager::addToList (lBomRoot, lBAInv);

  BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
  BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
                      "The inventory key, '" << lBAInv.describeKey()
                      << "', should be equal to '" << lBAAirlineCode
                      << "', but is not");

  // Create an Inventory for AF
  const stdair::AirlineCode_T lAFAirlineCode ("AF");
  const stdair::InventoryKey lAFKey (lAFAirlineCode);
  myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
  stdair::FacBomManager::addToList (lBomRoot, lAFInv);

  BOOST_CHECK_EQUAL (lAFInv.describeKey(), lAFAirlineCode);
  BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lAFAirlineCode,
                      "The inventory key, '" << lAFInv.describeKey()
                      << "', should be equal to '" << lAFAirlineCode
                      << "', but is not");

  // Browse the inventories
  const myprovider::InventoryList_T& lInventoryList =
      stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
  const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lAFAirlineCode};
  short idx = 0;
  for (myprovider::InventoryList_T::const_iterator itInv =
         lInventoryList.begin(); itInv != lInventoryList.end();
       ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
                         "They inventory key, '" << lInventoryKeyArray[idx]
                         << "', does not match that of the Inventory object: '"
                         << lInv_ptr->describeKey() << "'");
  }
}

BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {
```

```
// Backup (thanks to Boost.Serialisation) file
const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";

// Output log File
const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");

// Set the log parameters
std::ofstream logOutputFile;

// Open and clean the log outputfile
logOutputFile.open (lLogFilename.c_str());
logOutputFile.clear();

// Initialise the stdair BOM
const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
stdair::STDAIR_Service stdairService (lLogParams);

// Build a sample BOM tree
stdairService.buildSampleBom();

// Retrieve (a reference on) the top of the persistent BOM tree
stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

// DEBUG: Display the whole BOM tree
const std::string& lCSVDump = stdairService.csvDisplay ();
STDAIR_LOG_DEBUG (lCSVDump);

// Clone the persistent BOM
stdairService.clonePersistentBom ();

// Retrieve the BomRoot key, and compare it to the expected one
const std::string lBAInvKeyStr ("BA");
stdair::Inventory* lBAInv_ptr =
  lPersistentBomRoot.getInventory (lBAInvKeyStr);

// DEBUG
STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                  << lBAInvKeyStr << "' key.");

BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "'.");

// create and open a character archive for output
std::ofstream ofs (lBackupFilename.c_str());

// save data to archive
{
  boost::archive::text_oarchive oa (ofs);
  // write class instance to archive
  oa << lPersistentBomRoot;
  // archive and stream closed when destructors are called
}

// ... some time later restore the class instance to its orginal state
stdair::BomRoot& lRestoredBomRoot =
  stdair::FacBom<stdair::BomRoot>::instance().create();
{
  // create and open an archive for input
  std::ifstream ifs (lBackupFilename.c_str());
  boost::archive::text_iarchive ia(ifs);
  // read class state from archive
  ia >> lRestoredBomRoot;
  // archive and stream closed when destructors are called
}
```

```
  // DEBUG: Display the whole restored BOM tree
  const std::string& lRestoredCSVDump =
    stdairService.csvDisplay(lRestoredBomRoot);
  STDAIR_LOG_DEBUG (lRestoredCSVDump);

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
                       << "', should be equal to '" << lBomRootString
                       << "', but is not.");

  // Retrieve the Inventory
  stdair::Inventory* lRestoredBAInv_ptr =
    lRestoredBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the restored BOM root.");

  BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the restored BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_clone_test) {

  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_clone.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // Retrieve (a constant reference on) the top of the persistent BOM tree
  const stdair::BomRoot& lPersistentBomRoot =
    stdairService.getPersistentBomRoot();

  // DEBUG: Display the whole persistent BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG ("Display the persistent BOM tree.");
  STDAIR_LOG_DEBUG (lCSVDump);

  // Clone the persistent BOM
  stdairService.clonePersistentBom ();

  // Retrieve (a reference on) the top of the clone BOM tree
```

```
  stdair::BomRoot& lCloneBomRoot = stdairService.getBomRoot();

  // DEBUG: Display the clone BOM tree after the clone process.
  const std::string& lAfterCloneCSVDump =
    stdairService.csvDisplay(lCloneBomRoot);
  STDAIR_LOG_DEBUG ("Display the clone BOM tree after the clone process.");
  STDAIR_LOG_DEBUG (lAfterCloneCSVDump);

  // Retrieve the clone BomRoot key, and compare it to the persistent BomRoot
  // key.
  const std::string& lCloneBomRootKeyStr = lCloneBomRoot.describeKey();
  const std::string& lPersistentBomRootKeyStr =
    lPersistentBomRoot.describeKey();

  // DEBUG
  STDAIR_LOG_DEBUG ("The clone BOM root key is '" << lCloneBomRootKeyStr
                    << "'. It should be equal to '"
                    << lPersistentBomRootKeyStr << "'");

  BOOST_CHECK_EQUAL (lCloneBomRootKeyStr, lPersistentBomRootKeyStr);
  BOOST_CHECK_MESSAGE (lCloneBomRootKeyStr == lPersistentBomRootKeyStr,
                       "The clone BOM root key, '" << lCloneBomRootKeyStr
                       << "', should be equal to '" << lPersistentBomRootKeyStr
                       << "', but is not.");

  // Retrieve the BA inventory in the clone BOM root
  const std::string lBAInvKeyStr ("BA");
  stdair::Inventory* lCloneBAInv_ptr =
    lCloneBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the clone BOM root.");

  BOOST_CHECK_MESSAGE (lCloneBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the clone BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

#else // BOOST_VERSION >= 103900
boost_utf::test_suite* init_unit_test_suite (int, char* []) {
  boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
  return test;
}
#endif // BOOST_VERSION >= 103900

/*!
```

27 Namespace Index

27.1 Namespace List

Here is a list of all namespaces with brief descriptions:

28 Class Index

28.1 Class Hierarchy

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

29 Class Index

29.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

30 File Index

30.1 File List

Here is a list of all files with brief descriptions:

31 Namespace Documentation

31.1 boost Namespace Reference

Forward declarations.

Namespaces

- namespace serialization

31.1.1 Detailed Description

Forward declarations.

31.2 boost::serialization Namespace Reference

31.3 bpt Namespace Reference

Typedefs

- typedef char ptree

31.3.1 Typedef Documentation

31.3.1.1 typedef char bpt::ptree

Definition at line 22 of file BomINIImport.cpp.

31.4 soci Namespace Reference

Classes

- struct type_conversion< stdair::AirlineStruct >

31.5 stdair Namespace Reference

Handle on the StdAir library context.

Namespaces

- namespace LOG

Classes

- struct BasChronometer
- struct DefaultDCPList
- struct DefaultDtdFratMap
- struct DefaultDtdProbMap
- struct DefaultMap
- struct BasDBParams

  Structure holding the parameters for connection to a database.

- struct BasFileMgr
- struct BasLogParams

  Structure holding parameters for logging.

- struct date_time_element
- struct ContinuousAttributeLite

  Class modeling the distribution of values that can be taken by a continuous attribute.

- struct DemandGenerationMethod

  Enumeration of demand (booking request) generation methods.

- class DictionaryManager

  Class wrapper of dictionary business methods.

- struct EventType
- struct ForecastingMethod
- struct JSonCommand

  Enumeration of json commands.

- struct OptimisationMethod
- struct PartnershipTechnique

  Enumeration of partnership techniques.

- struct PassengerChoiceModel
- struct PassengerType
- struct PreOptimisationMethod
- struct ProgressStatus
- struct ProgressStatusSet
- struct RandomGeneration

  Class holding a random generator.

- struct SampleType

  Enumeration of BOM sample types.

- struct ServiceInitialisationType

  Enumeration of service initialisation types.

- struct StructAbstract

  Base class for the light structures.

- struct UnconstrainingMethod
- class YieldRange
- class AirlineClassList

  Class representing the actual attributes for a segment-features.

- struct AirlineClassListKey

  Key of airport-pair.

- class AirlineFeature

Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

- struct AirlineFeatureKey
- struct AirlineStruct
- class AirportPair

  Class representing the actual attributes for an airport-pair.

- struct AirportPairKey

  Key of airport-pair.

- class BomAbstract

  Base class for the Business Object Model (BOM) layer.

- class BomArchive

  Utility class to archive/restore BOM objects with Boost serialisation.

- class BomDisplay

  Utility class to display StdAir objects with a pretty format.

- class BomHolder

  Class representing the holder of BOM object containers (list and map).

- struct BomHolderKey
- struct BomID

  Class wrapper of bom ID (e.g. pointer to object).

- class BomINIImport

  Utility class to import StdAir objects in a INI format.

- class BomJSONExport

  Utility class to export StdAir objects in a JSON format.

- class BomJSONImport

  Utility class to import StdAir objects in a JSON format.

- class BomKeyManager

  Utility class to extract key structures from strings.

- class BomManager

  Utility class for StdAir-based objects.

- class BomRetriever

  Utility class to retrieve StdAir objects.

- class BomRoot

  Class representing the actual attributes for the Bom root.

- struct BomRootKey

  Key of the BOM structure root.

- class BookingClass
- struct BookingClassKey
- struct BookingRequestStruct

  Structure holding the elements of a booking request.

- struct BreakPointStruct
- class Bucket

  Class representing the actual attributes for an airline booking class.

- struct BucketKey

  Key of booking-class.

- struct CancellationStruct

  Structure holding the elements of a travel solution.

- struct ConfigHolderStruct
- class DatePeriod

  Class representing the actual attributes for a fare date-period.

- struct DatePeriodKey

  Key of date-period.

- struct DoWStruct
- struct EventStruct
- class FareFamily

  Class representing the actual attributes for a family fare.

- struct FareFamilyKey

  Key of a given fare family, made of a fare family code.

- class FareFeatures

  Class representing the actual attributes for a fare date-period.

- struct FareFeaturesKey

  Key of date-period.

- struct FareOptionStruct

  Structure holding the elements of a fare option.

- struct FFDisutilityCurveHolderStruct
- class FlightDate

  Class representing the actual attributes for an airline flight-date.

- struct FlightDateKey

  Key of a given flight-date, made of a flight number and a departure date.

- class FlightPeriod
- struct FlightPeriodKey
- struct FRAT5CurveHolderStruct

- class Inventory

  Class representing the actual attributes for an airline inventory.

- struct InventoryKey

  Key of a given inventory, made of the airline code.

- struct KeyAbstract

  Base class for the keys of Business Object Model (BOM) layer.

- class LegCabin

  Class representing the actual attributes for an airline leg-cabin.

- struct LegCabinKey

  Key of a given leg-cabin, made of a cabin code (only).

- class LegDate
- struct LegDateKey
- class NestingNode
- struct NestingNodeKey

  Key of a given policy, made of a policy code.

- struct NestingStructureKey

  Key of a given policy, made of a policy code.

- class OnDDate

  Class representing the actual attributes for an airline flight-date.

- struct OnDDateKey

  Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

- struct OptimisationNotificationStruct
- struct ParsedKey
- struct PeriodStruct
- class Policy
- struct PolicyKey

  Key of a given policy, made of a policy code.

- class PosChannel

  Class representing the actual attributes for a fare point of sale.

- struct PosChannelKey

  Key of point of sale and channel.

- struct RMEventStruct
- class SegmentCabin

  Class representing the actual attributes for an airline segment-cabin.

- struct SegmentCabinKey

  Key of a given segment-cabin, made of a cabin code (only).

- class SegmentDate

  Class representing the actual attributes for an airline segment-date.

- struct SegmentDateKey

  Key of a given segment-date, made of an origin and a destination airports.

- class SegmentPeriod
- struct SegmentPeriodKey
- class SegmentSnapshotTable

  Class representing the actual attributes for an airline segment data tables.

- struct SegmentSnapshotTableKey

  Key of a given guillotine block, made of a guillotine number.

- class SimpleNestingStructure
- struct SnapshotStruct
- class TimePeriod

  Class representing the actual attributes for a fare time-period.

- struct TimePeriodKey

  Key of time-period.

- struct TravelSolutionStruct

  Structure holding the elements of a travel solution.

- struct VirtualClassStruct
- class YieldFeatures

  Class representing the actual attributes for a yield date-period.

- struct YieldFeaturesKey

  Key of date-period.

- class YieldStore
- struct YieldStoreKey
- class CmdAbstract
- class CmdBomManager
- class CmdBomSerialiser
- class CmdCloneBomManager
- class DBManagerForAirlines
- class DbaAbstract
- class FacAbstract
- class FacBom

  Base class for Factory layer.

- class FacBomManager

  Utility class for linking StdAir-based objects.

- class FacCloneBom

  Base class for Factory layer.

- class DBSessionManager
- class FacServiceAbstract
- class FacSTDAIRServiceContext

  Factory for Bucket.

- class FacSupervisor
- class Logger
- class ServiceAbstract
- class STDAIR_ServiceContext

  Class holding the context of the Stdair services.

- class RootException

  Root of the stdair exceptions.

- class FileNotFoundException
- class NonInitialisedLogServiceException
- class NonInitialisedServiceException
- class NonInitialisedContainerException
- class NonInitialisedRelationShipException
- class MemoryAllocationException
- class ObjectLinkingException
- class DocumentNotFoundException
- class ParserException
- class SerialisationException
- class KeyNotFoundException
- class CodeConversionException
- class CodeDuplicationException
- class KeyDuplicationException
- class ObjectCreationgDuplicationException
- class ObjectNotFoundException
- class ParsingFileFailedException
- class SQLDatabaseException
- class NonInitialisedDBSessionManagerException
- class SQLDatabaseConnectionImpossibleException
- class EventException
- class SimpleNestingStructException
- class BookingClassListEmptyInNestingStructException
- class RootFilePath

  Root of the input and output files.

- class InputFilePath
- class ScheduleFilePath
- class ODFilePath
- class FRAT5FilePath
- class FFDisutilityFilePath
- class ConfigINIFile
- class JSONString

  JSON-formatted string.

- class STDAIR_Service

  Interface for the STDAIR Services.

Typedefs

- typedef date_time_element< 0, 23 > hour_t
- typedef date_time_element< 0, 59 > minute_t
- typedef date_time_element< 0, 59 > second_t
- typedef date_time_element< 1900, 2100 > year_t
- typedef date_time_element< 1, 12 > month_t
- typedef date_time_element< 1, 31 > day_t
- typedef std::istreambuf_iterator< char > base_iterator_t
- typedef boost::spirit::multi_pass< base_iterator_t > iterator_t
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > int1_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > uint2_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > uint4_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > uint1_4_p_t
- typedef boost::spirit::qi::uint_parser< hour_t, 10, 2, 2 > hour_p_t
- typedef boost::spirit::qi::uint_parser< minute_t, 10, 2, 2 > minute_p_t
- typedef boost::spirit::qi::uint_parser< second_t, 10, 2, 2 > second_p_t
- typedef boost::spirit::qi::uint_parser< year_t, 10, 4, 4 > year_p_t
- typedef boost::spirit::qi::uint_parser< month_t, 10, 2, 2 > month_p_t
- typedef boost::spirit::qi::uint_parser< day_t, 10, 2, 2 > day_p_t
- typedef unsigned short DictionaryKey_T
- typedef std::list< AirlineClassList * > AirlineClassListList_T
- typedef std::map< const MapKey_T, AirlineClassList * > AirlineClassListMap_T
- typedef std::pair< MapKey_T, AirlineClassList * > AirlineClassListWithKey_T
- typedef std::list< AirlineClassListWithKey_T > AirlineClassListDetailedList_T
- typedef std::list< AirlineFeature * > AirlineFeatureList_T
- typedef std::map< const MapKey_T, AirlineFeature * > AirlineFeatureMap_T
- typedef std::list< AirportPair * > AirportPairList_T
- typedef std::map< const MapKey_T, AirportPair * > AirportPairMap_T
- typedef std::pair< MapKey_T, AirportPair * > AirportPairWithKey_T
- typedef std::list< AirportPairWithKey_T > AirportPairDetailedList_T
- typedef std::map< const std::type_info *, BomAbstract * > HolderMap_T
- typedef struct BomID< BookingClass > BookingClassID_T
- typedef std::list< BookingClassID_T > BookingClassIDList_T
- typedef boost::tokenizer< boost::char_separator< char > > Tokeniser_T
- typedef std::list< BookingClass * > BookingClassList_T
- typedef std::map< const MapKey_T, BookingClass * > BookingClassMap_T
- typedef boost::shared_ptr< BookingRequestStruct > BookingRequestPtr_T
- typedef std::string DemandGeneratorKey_T
- typedef boost::shared_ptr< BreakPointStruct > BreakPointPtr_T
- typedef std::list< BreakPointStruct > BreakPointList_T
- typedef std::list< Bucket * > BucketList_T
- typedef std::map< const MapKey_T, Bucket * > BucketMap_T
- typedef boost::shared_ptr< CancellationStruct > CancellationPtr_T
- typedef boost::shared_ptr< ConfigHolderStruct > ConfigHolderPtr_T
- typedef std::list< DatePeriod * > DatePeriodList_T
- typedef std::map< const MapKey_T, DatePeriod * > DatePeriodMap_T
- typedef std::pair< MapKey_T, DatePeriod * > DatePeriodWithKey_T
- typedef std::list< DatePeriodWithKey_T > DatePeriodDetailedList_T
- typedef std::pair< const LongDuration_T, EventStruct > EventListElement_T

- typedef std::map< const LongDuration_T, EventStruct > EventList_T
- typedef std::list< FareFamily * > FareFamilyList_T
- typedef std::map< const MapKey_T, FareFamily * > FareFamilyMap_T
- typedef std::list< FareFeatures * > FareFeaturesList_T
- typedef std::map< const MapKey_T, FareFeatures * > FareFeaturesMap_T
- typedef std::pair< MapKey_T, FareFeatures * > FareFeaturesWithKey_T
- typedef std::list< FareFeaturesWithKey_T > FareFeaturesDetailedList_T
- typedef std::list< FareOptionStruct > FareOptionList_T
- typedef std::map< const std::string, FFDisutilityCurve_T > FFDisutilityCurveHolder_T
- typedef std::list< FlightDate * > FlightDateList_T
- typedef std::map< const MapKey_T, FlightDate * > FlightDateMap_T
- typedef std::list< FlightPeriod * > FlightPeriodList_T
- typedef std::map< const MapKey_T, FlightPeriod * > FlightPeriodMap_T
- typedef std::map< const std::string, FRAT5Curve_T > FRAT5CurveHolder_T
- typedef std::list< Inventory * > InventoryList_T
- typedef std::map< const MapKey_T, Inventory * > InventoryMap_T
- typedef std::string MapKey_T
- typedef std::list< std::string > KeyList_T
- typedef std::list< LegCabin * > LegCabinList_T
- typedef std::map< const MapKey_T, LegCabin * > LegCabinMap_T
- typedef std::list< LegDate * > LegDateList_T
- typedef std::map< const MapKey_T, LegDate * > LegDateMap_T
- typedef std::list< NestingNode * > NestingNodeList_T
- typedef std::map< const MapKey_T, NestingNode * > NestingNodeMap_T
- typedef std::list< OnDDate * > OnDDateList_T
- typedef std::map< const MapKey_T, OnDDate * > OnDDateMap_T
- typedef std::pair< std::string, YieldDemandPair_T > StringDemandStructPair_T
- typedef std::map< std::string, YieldDemandPair_T > StringDemandStructMap_T
- typedef std::map< std::string, CabinClassPairList_T > StringCabinClassPairListMap_T
- typedef std::pair< std::string, CabinClassPairList_T > StringCabinClassPair_T
- typedef std::map< CabinCode_T, WTPDemandPair_T > CabinForecastMap_T
- typedef std::pair< CabinCode_T, WTPDemandPair_T > CabinForecastPair_T
- typedef boost::shared_ptr< OptimisationNotificationStruct > OptimisationNotificationPtr_T
- typedef std::list< Policy * > PolicyList_T
- typedef std::map< const MapKey_T, Policy * > PolicyMap_T
- typedef std::list< PosChannel * > PosChannelList_T
- typedef std::map< const MapKey_T, PosChannel * > PosChannelMap_T
- typedef std::pair< MapKey_T, PosChannel * > PosChannelWithKey_T
- typedef std::list< PosChannelWithKey_T > PosChannelDetailedList_T
- typedef boost::shared_ptr< RMEventStruct > RMEventPtr_T
- typedef std::list< RMEventStruct > RMEventList_T
- typedef std::list< SegmentCabin * > SegmentCabinList_T
- typedef std::map< const MapKey_T, SegmentCabin * > SegmentCabinMap_T
- typedef std::list< std::string > RoutingLegKeyList_T
- typedef std::list< SegmentDate * > SegmentDateList_T
- typedef std::map< const MapKey_T, SegmentDate * > SegmentDateMap_T
- typedef std::list< SegmentPeriod * > SegmentPeriodList_T
- typedef std::map< const MapKey_T, SegmentPeriod * > SegmentPeriodMap_T
- typedef std::pair< MapKey_T, SegmentPeriod * > SegmentPeriodWithKey_T
- typedef std::list< SegmentPeriodWithKey_T > SegmentPeriodDetailedList_T

- typedef std::list< SegmentSnapshotTable * > SegmentSnapshotTableList_T
- typedef std::map< const MapKey_T, SegmentSnapshotTable * > SegmentSnapshotTableMap_T
- typedef std::map< const SegmentCabin *, SegmentDataID_T > SegmentCabinIndexMap_T
- typedef std::map< const MapKey_T, ClassIndex_T > ClassIndexMap_T
- typedef std::list< SimpleNestingStructure * > SimpleNestingStructureList_T
- typedef std::map< const MapKey_T, SimpleNestingStructure * > SimpleNestingStructureMap_T
- typedef boost::shared_ptr< SnapshotStruct > SnapshotPtr_T
- typedef std::list< TimePeriod * > TimePeriodList_T
- typedef std::map< const MapKey_T, TimePeriod * > TimePeriodMap_T
- typedef std::pair< MapKey_T, TimePeriod * > TimePeriodWithKey_T
- typedef std::list< TimePeriodWithKey_T > TimePeriodDetailedList_T
- typedef std::list< TravelSolutionStruct > TravelSolutionList_T
- typedef KeyList_T SegmentPath_T
- typedef std::list< SegmentPath_T > SegmentPathList_T
- typedef std::map< const ClassCode_T, Availability_T > ClassAvailabilityMap_T
- typedef std::list< ClassAvailabilityMap_T > ClassAvailabilityMapHolder_T
- typedef std::map< const ClassCode_T, BookingClassID_T > ClassObjectIDMap_T
- typedef std::list< ClassObjectIDMap_T > ClassObjectIDMapHolder_T
- typedef std::map< const ClassCode_T, YieldValue_T > ClassYieldMap_T
- typedef std::list< ClassYieldMap_T > ClassYieldMapHolder_T
- typedef std::list< BidPriceVector_T > BidPriceVectorHolder_T
- typedef std::map< const ClassCode_T, const BidPriceVector_T * > ClassBpvMap_T
- typedef std::list< ClassBpvMap_T > ClassBpvMapHolder_T
- typedef std::list< VirtualClassStruct > VirtualClassList_T
- typedef std::map< const YieldLevel_T, VirtualClassStruct > VirtualClassMap_T
- typedef std::list< YieldFeatures * > YieldFeaturesList_T
- typedef std::map< const MapKey_T, YieldFeatures * > YieldFeaturesMap_T
- typedef std::pair< MapKey_T, YieldFeatures * > YieldFeaturesWithKey_T
- typedef std::list< YieldFeaturesWithKey_T > YieldFeaturesDetailedList_T
- typedef std::list< YieldStore * > YieldStoreList_T
- typedef std::map< const MapKey_T, YieldStore * > YieldStoreMap_T
- typedef std::string LocationCode_T
- typedef unsigned long int Distance_T
- typedef LocationCode_T AirportCode_T
- typedef LocationCode_T CityCode_T
- typedef std::string KeyDescription_T
- typedef std::string AirlineCode_T
- typedef unsigned short FlightNumber_T
- typedef unsigned short TableID_T
- typedef std::string CabinCode_T
- typedef std::string FamilyCode_T
- typedef std::string PolicyCode_T
- typedef std::string NestingStructureCode_T
- typedef std::string NestingNodeCode_T
- typedef std::string ClassCode_T
- typedef unsigned long Identity_T
- typedef std::string TripType_T
- typedef double MonetaryValue_T
- typedef double RealNumber_T
- typedef double Percentage_T

- typedef double PriceValue_T
- typedef double YieldValue_T
- typedef std::string PriceCurrency_T
- typedef double Revenue_T
- typedef double Multiplier_T
- typedef double NbOfSeats_T
- typedef unsigned int Count_T
- typedef short PartySize_T
- typedef double NbOfRequests_T
- typedef NbOfRequests_T NbOfBookings_T
- typedef NbOfRequests_T NbOfCancellations_T
- typedef unsigned short NbOfTravelSolutions_T
- typedef std::string ClassList_String_T
- typedef unsigned short NbOfSegments_T
- typedef unsigned short NbOfAirlines_T
- typedef double Availability_T
- typedef double Fare_T
- typedef bool Flag_T
- typedef unsigned int UnsignedIndex_T
- typedef unsigned int NbOfClasses_T
- typedef unsigned int NbOfFareFamilies_T
- typedef std::string Filename_T
- typedef std::string FileAddress_T
- typedef float ProgressPercentage_T
- typedef boost::posix_time::time_duration Duration_T
- typedef boost::gregorian::date Date_T
- typedef boost::posix_time::time_duration Time_T
- typedef boost::posix_time::ptime DateTime_T
- typedef boost::gregorian::date_period DatePeriod_T
- typedef std::string DOW_String_T
- typedef boost::gregorian::date_duration DateOffset_T
- typedef int DayDuration_T
- typedef bool SaturdayStay_T
- typedef long int IntDuration_T
- typedef long long int LongDuration_T
- typedef float FloatDuration_T
- typedef soci::session DBSession_T
- typedef soci::statement DBRequestStatement_T
- typedef std::string DBConnectionName_T
- typedef bool ChangeFees_T
- typedef bool NonRefundable_T
- typedef double SaturdayStayRatio_T
- typedef double ChangeFeesRatio_T
- typedef double NonRefundableRatio_T
- typedef double Disutility_T
- typedef std::string PassengerType_T
- typedef std::string DistributionPatternId_T
- typedef std::string CancellationRateCurveId_T
- typedef std::string AirlinePreferenceId_T
- typedef std::pair< Percentage_T, Percentage_T > CancellationNoShowRatePair_T

- typedef std::string CharacteristicsPatternId_T
- typedef std::string CharacteristicsIndex_T
- typedef double WTP_T
- typedef boost::tuples::tuple< double, WTP_T > CharacteristicsWTP_tuple_T
- typedef std::pair< WTP_T, MeanStdDevPair_T > WTPDemandPair_T
- typedef NbOfRequests_T NbOfNoShows_T
- typedef double MatchingIndicator_T
- typedef std::string DemandStreamKeyStr_T
- typedef std::string ChannelLabel_T
- typedef std::string FrequentFlyer_T
- typedef std::string RequestStatus_T
- typedef std::map< Identity_T, Identity_T > BookingTSIDMap_T
- typedef std::pair< CabinCode_T, ClassCode_T > CabinClassPair_T
- typedef std::list< CabinClassPair_T > CabinClassPairList_T
- typedef double ProportionFactor_T
- typedef std::list< ProportionFactor_T > ProportionFactorList_T
- typedef std::string OnDString_T
- typedef std::list< OnDString_T > OnDStringList_T
- typedef std::string EventName_T
- typedef double NbOfEvents_T
- typedef std::string EventGeneratorKey_T
- typedef double NbOfFareRules_T
- typedef std::string NetworkID_T
- typedef std::vector< AirlineCode_T > AirlineCodeList_T
- typedef std::vector< ClassList_String_T > ClassList_StringList_T
- typedef std::vector< ClassCode_T > ClassCodeList_T
- typedef unsigned short SubclassCode_T
- typedef std::string FlightPathCode_T
- typedef std::map< CabinCode_T, ClassList_String_T > CabinBookingClassMap_T
- typedef std::string CurveKey_T
- typedef double CabinCapacity_T
- typedef double NbOfFlightDates_T
- typedef double CommittedSpace_T
- typedef double UPR_T
- typedef double BookingLimit_T
- typedef double AuthorizationLevel_T
- typedef double CapacityAdjustment_T
- typedef double BlockSpace_T
- typedef bool AvailabilityStatus_T
- typedef std::vector< Availability_T > BucketAvailabilities_T
- typedef double NbOfYields_T
- typedef double NbOfInventoryControlRules_T
- typedef bool CensorshipFlag_T
- typedef short DTD_T
- typedef short DCP_T
- typedef std::list< DCP_T > DCPList_T
- typedef std::map< DTD_T, RealNumber_T > DTDFratMap_T
- typedef std::map< FloatDuration_T, float > DTDProbMap_T
- typedef std::vector< CensorshipFlag_T > CensorshipFlagList_T
- typedef double BookingRatio_T

- typedef double Yield_T
- typedef unsigned int YieldLevel_T
- typedef std::map< YieldLevel_T, MeanStdDevPair_T > YieldLevelDemandMap_T
- typedef std::pair< Yield_T, MeanStdDevPair_T > YieldDemandPair_T
- typedef double BidPrice_T
- typedef std::vector< BidPrice_T > BidPriceVector_T
- typedef unsigned int SeatIndex_T
- typedef std::string ControlMode_T
- typedef double OverbookingRate_T
- typedef double ProtectionLevel_T
- typedef std::vector< double > EmsrValueList_T
- typedef std::vector< double > BookingLimitVector_T
- typedef std::vector< double > ProtectionLevelVector_T
- typedef boost::multi_array< double, 2 > SnapshotBlock_T
- typedef SnapshotBlock_T::index_range SnapshotBlockRange_T
- typedef SnapshotBlock_T::array_view< 1 >::type SegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::array_view< 2 >::type SegmentCabinDTDRangeSnapshotView_T
- typedef SnapshotBlock_T::const_array_view< 1 >::type ConstSegmentCabinDTDSnapshotView_-T
- typedef SnapshotBlock_T::const_array_view< 2 >::type ConstSegmentCabinDTDRangeSnapshotView_-T
- typedef unsigned short SegmentDataID_T
- typedef unsigned short LegDataID_T
- typedef unsigned short ClassIndex_T
- typedef unsigned int ReplicationNumber_T
- typedef unsigned long int ExponentialSeed_T
- typedef unsigned long int UniformSeed_T
- typedef unsigned long int RandomSeed_T
- typedef boost::minstd_rand BaseGenerator_T
- typedef boost::uniform_real UniformDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, UniformDistribution_T > UniformGenerator_T
- typedef boost::normal_distribution NormalDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, NormalDistribution_T > NormalGenerator_T
- typedef boost::exponential_distribution ExponentialDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, ExponentialDistribution_T > ExponentialGenerator_T
- typedef double MeanValue_T
- typedef double StdDevValue_T
- typedef std::pair< MeanValue_T, StdDevValue_T > MeanStdDevPair_T
- typedef std::vector< MeanStdDevPair_T > MeanStdDevPairVector_T
- typedef float Probability_T
- typedef std::string ForecasterMode_T
- typedef short HistoricalDataLimit_T
- typedef std::string OptimizerMode_T
- typedef NbOfBookings_T PolicyDemand_T
- typedef std::vector< double > GeneratedDemandVector_T
- typedef std::vector< GeneratedDemandVector_T > GeneratedDemandVectorHolder_T
- typedef double SellupProbability_T

- typedef std::vector< NbOfRequests_T > UncDemVector_T
- typedef std::vector< NbOfBookings_T > BookingVector_T
- typedef double FRAT5_T
- typedef std::map< const DTD_T, FRAT5_T > FRAT5Curve_T
- typedef std::map< const DTD_T, double > FFDisutilityCurve_T
- typedef std::map< const DTD_T, double > SellUpCurve_T
- typedef std::map< const DTD_T, double > DispatchingCurve_T
- typedef std::map< BookingClass *, SellUpCurve_T > BookingClassSellUpCurveMap_T
- typedef std::map< BookingClass *, DispatchingCurve_T > BookingClassDispatchingCurveMap_T
- typedef std::map< const Yield_T, double > YieldDemandMap_T
- typedef unsigned int NbOfSamples_T
- typedef boost::shared_ptr< STDAIR_Service > STDAIR_ServicePtr_T

Functions

- const std::string DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ")
- const double DEFAULT_EPSILON_VALUE (0.0001)
- const unsigned int DEFAULT_FLIGHT_SPEED (900)
- const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0)
- const Duration_T NULL_BOOST_TIME_DURATION (-1,-1,-1)
- const Duration_T DEFAULT_NULL_DURATION (0, 0, 0)
- const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)
- const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)
- const DayDuration_T DEFAULT_DAY_DURATION (0)
- const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T DEFAULT_DOW_STRING ("0000000")
- const DateOffset_T DEFAULT_DATE_OFFSET (0)
- const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1)
- const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_-DURATION)
- const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1)
- const Count_T SECONDS_IN_ONE_DAY (86400)
- const Count_T MILLISECONDS_IN_ONE_SECOND (1000)
- const RandomSeed_T DEFAULT_RANDOM_SEED (120765987)
- const AirportCode_T AIRPORT_LHR ("LHR")
- const AirportCode_T AIRPORT_SYD ("SYD")
- const CityCode_T POS_LHR ("LHR")
- const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15)
- const Date_T DATE_20111231 (2011, boost::gregorian::Dec, 31)
- const DayDuration_T NO_ADVANCE_PURCHASE (0)
- const SaturdayStay_T SATURDAY_STAY (true)
- const SaturdayStay_T NO_SATURDAY_STAY (false)
- const ChangeFees_T CHANGE_FEES (true)
- const ChangeFees_T NO_CHANGE_FEES (false)
- const NonRefundable_T NON_REFUNDABLE (true)
- const NonRefundable_T NO_NON_REFUNDABLE (false)
- const SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY (true)
- const ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES (true)
- const NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE (true)
- const DayDuration_T NO_STAY_DURATION (0)

- const AirlineCode_T AIRLINE_CODE_BA ("BA")
- const CabinCode_T CABIN_Y ("Y")
- const ClassCode_T CLASS_CODE_Y ("Y")
- const ClassCode_T CLASS_CODE_Q ("Q")
- const AirportCode_T AIRPORT_SIN ("SIN")
- const AirportCode_T AIRPORT_BKK ("BKK")
- const CityCode_T POS_SIN ("SIN")
- const CabinCode_T CABIN_ECO ("Eco")
- const FrequentFlyer_T FREQUENT_FLYER_MEMBER ("M")
- const FamilyCode_T DEFAULT_FAMILY_CODE ("0")
- const PolicyCode_T DEFAULT_POLICY_CODE ("0")
- const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")
- const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")
- const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting")
- const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE ("0")
- const NbOfAirlines_T DEFAULT_NBOFAIRLINES (0)
- const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE ("")
- const Distance_T DEFAULT_DISTANCE_VALUE (0)
- const ClassCode_T DEFAULT_CLOSED_CLASS_CODE ("CC")
- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS (0)
- const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)
- const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION (0)
- const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)
- const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS (0)
- const CabinCapacity_T DEFAULT_CABIN_CAPACITY (100.0)
- const CommittedSpace_T DEFAULT_COMMITTED_SPACE (0.0)
- const BlockSpace_T DEFAULT_BLOCK_SPACE (0.0)
- const Availability_T DEFAULT_NULL_AVAILABILITY (0.0)
- const Availability_T DEFAULT_AVAILABILITY (9.0)
- const Availability_T MAXIMAL_AVAILABILITY (9999.0)
- const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG (false)
- const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0)
- const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE (0.0)
- const BookingRatio_T DEFAULT_OND_BOOKING_RATE (0.0)
- const Fare_T DEFAULT_FARE_VALUE (0.0)
- const Yield_T DEFAULT_CLASS_YIELD_VALUE (0.0)
- const Revenue_T DEFAULT_REVENUE_VALUE (0.0)
- const Percentage_T DEFAULT_LOAD_FACTOR_VALUE (100.0)
- const Yield_T DEFAULT_YIELD_VALUE (0.0)
- const Yield_T DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >::max())
- const NbOfBookings_T DEFAULT_YIELD_NB_OF_BOOKINGS (0.0)
- const Identity_T DEFAULT_BOOKING_NUMBER (0)
- const NbOfCancellations_T DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0)

- const NbOfNoShows_T DEFAULT_YIELD_NB_OF_NOSHOWS (0.0)
- const Availability_T DEFAULT_YIELD_AVAILABILITY (0.0)
- const CensorshipFlag_T DEFAULT_YIELD_CENSORSHIPFLAG (false)
- const BookingLimit_T DEFAULT_YIELD_BOOKING_LIMIT (0.0)
- const OverbookingRate_T DEFAULT_YIELD_OVERBOOKING_RATE (0.0)
- const Fare_T DEFAULT_OND_FARE_VALUE (0.0)
- const Count_T DEFAULT_PROGRESS_STATUS (0)
- const Percentage_T MAXIMUM_PROGRESS_STATUS (100)
- const Date_T DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)
- const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_-DATE, NULL_BOOST_TIME_DURATION)
- const PartySize_T DEFAULT_PARTY_SIZE (1)
- const DayDuration_T DEFAULT_STAY_DURATION (7)
- const WTP_T DEFAULT_WTP (1000.0)
- const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE)
- const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)
- const DateOffset_T DEFAULT_ADVANCE_PURCHASE (22)
- const Date_T DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_DATE-DEFAULT_ADVANCE_PURCHASE)
- const Duration_T DEFAULT_REQUEST_TIME (8, 0, 0)
- const DateTime_T DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const CabinCode_T DEFAULT_PREFERRED_CABIN ("M")
- const CityCode_T DEFAULT_POS ("ALL")
- const ChannelLabel_T DEFAULT_CHANNEL ("DC")
- const ChannelLabel_T CHANNEL_DN ("DN")
- const ChannelLabel_T CHANNEL_IN ("IN")
- const TripType_T TRIP_TYPE_ONE_WAY ("OW")
- const TripType_T TRIP_TYPE_ROUND_TRIP ("RT")
- const TripType_T TRIP_TYPE_INBOUND ("RI")
- const TripType_T TRIP_TYPE_OUTBOUND ("RO")
- const FrequentFlyer_T DEFAULT_FF_TIER ("N")
- const PriceValue_T DEFAULT_VALUE_OF_TIME (100.0)
- const IntDuration_T HOUR_CONVERTED_IN_SECONDS (3600)
- const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)
- const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)
- const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR (0.0)
- const PriceCurrency_T DEFAULT_CURRENCY ("EUR")
- const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS (false)
- const AirlineCode_T DEFAULT_AIRLINE_CODE ("XX")
- const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE ("")
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER (9999)
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF (255)
- const TableID_T DEFAULT_TABLE_ID (9999)
- const Date_T DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)
- const AirportCode_T DEFAULT_AIRPORT_CODE ("XXX")
- const AirportCode_T DEFAULT_NULL_AIRPORT_CODE ("")
- const AirportCode_T DEFAULT_ORIGIN ("XXX")
- const AirportCode_T DEFAULT_DESTINATION ("YYY")
- const CabinCode_T DEFAULT_CABIN_CODE ("X")

- const FamilyCode_T DEFAULT_FARE_FAMILY_CODE ("EcoSaver")
- const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")
- const ClassCode_T DEFAULT_CLASS_CODE ("X")
- const ClassCode_T DEFAULT_NULL_CLASS_CODE ("")
- const BidPrice_T DEFAULT_BID_PRICE (0.0)
- const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)
- const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)
- const SeatIndex_T DEFAULT_SEAT_INDEX (1)
- const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER (0)
- const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)
- const UPR_T DEFAULT_NULL_UPR (0)
- const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")
- const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")
- const std::string DEFAULT_KEY_FLD_DELIMITER (";")
- const std::string DEFAULT_KEY_SUB_FLD_DELIMITER (",")
- const boost::char_separator< char > DEFAULT_KEY_TOKEN_DELIMITER (";, ")
- template<int MIN, int MAX>
  date_time_element< MIN, MAX > operator* (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX>
  date_time_element< MIN, MAX > operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template void AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- void intDisplay (std::ostream &oStream, const int &iInt)
- template void BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- const boost::char_separator< char > TokeniserDashSeparator ("-")
- const boost::char_separator< char > TokeniserTimeSeparator (":")
- template void PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

- template void SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 >
  void serialiseHelper (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)
- template void BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

Variables

- const std::string DOW_STR [ ]
- const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD ('E')
- const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE ('N')
- const ForecastingMethod DEFAULT_FORECASTING_METHOD ('Q')
- const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD ('N')
- const OptimisationMethod DEFAULT_OPTIMISATION_METHOD ('M')
- const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const Date_T DEFAULT_DICO_STUDIED_DATE
- const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST
- const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST
- const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()
- const int DEFAULT_MAX_DTD = 365
- const DCPList_T DEFAULT_DCP_LIST = DefaultDCPList::init()
- const FRAT5Curve_T FRAT5_CURVE_A
- const FRAT5Curve_T FRAT5_CURVE_B
- const FRAT5Curve_T FRAT5_CURVE_C
- const FRAT5Curve_T FRAT5_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F
- const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T DEFAULT_DTD_PROB_MAP
- const OnDStringList_T DEFAULT_OND_STRING_LIST
- const std::string DISPLAY_LEVEL_STRING_ARRAY [51]
- const std::string DEFAULT_KEY_FLD_DELIMITER

- const std::string DEFAULT_KEY_SUB_FLD_DELIMITER
- const boost::char_separator< char > DEFAULT_KEY_TOKEN_DELIMITER
- const Distance_T DEFAULT_DISTANCE_VALUE
- const ClassCode_T DEFAULT_CLOSED_CLASS_CODE
- const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS
- const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
- const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN
- const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION
- const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS
- const CabinCapacity_T DEFAULT_CABIN_CAPACITY
- const CommittedSpace_T DEFAULT_COMMITTED_SPACE
- const BlockSpace_T DEFAULT_BLOCK_SPACE
- const Availability_T DEFAULT_NULL_AVAILABILITY
- const Availability_T DEFAULT_AVAILABILITY
- const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG
- const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT
- const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE
- const Fare_T DEFAULT_FARE_VALUE
- const Revenue_T DEFAULT_REVENUE_VALUE
- const PriceCurrency_T DEFAULT_CURRENCY
- const Percentage_T DEFAULT_LOAD_FACTOR_VALUE
- const DayDuration_T DEFAULT_DAY_DURATION
- const double DEFAULT_EPSILON_VALUE
- const AirportCode_T AIRPORT_LHR
- const AirportCode_T AIRPORT_SYD
- const CityCode_T POS_LHR
- const DayDuration_T NO_ADVANCE_PURCHASE
- const SaturdayStay_T SATURDAY_STAY
- const SaturdayStay_T NO_SATURDAY_STAY
- const ChangeFees_T CHANGE_FEES
- const ChangeFees_T NO_CHANGE_FEES
- const NonRefundable_T NON_REFUNDABLE
- const NonRefundable_T NO_NON_REFUNDABLE
- const DayDuration_T NO_STAY_DURATION
- const CabinCode_T CABIN_Y
- const AirlineCode_T AIRLINE_CODE_BA
- const ClassCode_T CLASS_CODE_Y
- const ClassCode_T CLASS_CODE_Q
- const AirportCode_T AIRPORT_SIN
- const AirportCode_T AIRPORT_BKK
- const CityCode_T POS_SIN
- const CabinCode_T CABIN_ECO
- const FrequentFlyer_T FREQUENT_FLYER_MEMBER
- const Count_T DEFAULT_PROGRESS_STATUS

- const Date_T DEFAULT_EVENT_OLDEST_DATE
- const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME
- const Percentage_T MAXIMUM_PROGRESS_STATUS
- const std::string DEFAULT_BOM_ROOT_KEY
- const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES
- const unsigned int DEFAULT_FLIGHT_SPEED
- const BookingRatio_T DEFAULT_OND_BOOKING_RATE
- const Count_T SECONDS_IN_ONE_DAY
- const Count_T MILLISECONDS_IN_ONE_SECOND
- const Date_T DEFAULT_DATE
- const DateTime_T DEFAULT_DATETIME
- const Duration_T DEFAULT_EPSILON_DURATION
- const RandomSeed_T DEFAULT_RANDOM_SEED
- const Duration_T NULL_BOOST_TIME_DURATION
- const Duration_T DEFAULT_NULL_DURATION
- const Fare_T DEFAULT_CLASS_FARE_VALUE
- const NbOfAirlines_T DEFAULT_NBOFAIRLINES
- const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR
- const ChannelLabel_T DEFAULT_CHANNEL
- const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS
- const AirlineCode_T DEFAULT_AIRLINE_CODE
- const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER
- const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF
- const TableID_T DEFAULT_TABLE_ID
- const Date_T DEFAULT_DEPARTURE_DATE
- const AirportCode_T DEFAULT_AIRPORT_CODE
- const AirportCode_T DEFAULT_NULL_AIRPORT_CODE
- const AirportCode_T DEFAULT_ORIGIN
- const AirportCode_T DEFAULT_DESTINATION
- const CabinCode_T DEFAULT_CABIN_CODE
- const FamilyCode_T DEFAULT_FARE_FAMILY_CODE
- const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE
- const PolicyCode_T DEFAULT_POLICY_CODE
- const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE
- const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE
- const ClassCode_T DEFAULT_CLASS_CODE
- const ClassCode_T DEFAULT_NULL_CLASS_CODE
- const BidPrice_T DEFAULT_BID_PRICE
- const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT
- const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND
- const Availability_T MAXIMAL_AVAILABILITY
- const SeatIndex_T DEFAULT_SEAT_INDEX
- const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER
- const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT
- const UPR_T DEFAULT_NULL_UPR
- const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE
- const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE

- const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD
- const DOW_String_T DEFAULT_DOW_STRING
- const DateOffset_T DEFAULT_DATE_OFFSET
- const PartySize_T DEFAULT_PARTY_SIZE
- const DayDuration_T DEFAULT_STAY_DURATION
- const WTP_T DEFAULT_WTP
- const CityCode_T DEFAULT_POS
- const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE
- const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME
- const DateOffset_T DEFAULT_ADVANCE_PURCHASE
- const Date_T DEFAULT_REQUEST_DATE
- const Duration_T DEFAULT_REQUEST_TIME
- const DateTime_T DEFAULT_REQUEST_DATE_TIME
- const CabinCode_T DEFAULT_PREFERRED_CABIN
- const ChannelLabel_T CHANNEL_DN
- const ChannelLabel_T CHANNEL_IN
- const TripType_T TRIP_TYPE_ONE_WAY
- const TripType_T TRIP_TYPE_ROUND_TRIP
- const TripType_T TRIP_TYPE_INBOUND
- const TripType_T TRIP_TYPE_OUTBOUND
- const FrequentFlyer_T DEFAULT_FF_TIER
- const PriceValue_T DEFAULT_VALUE_OF_TIME
- const IntDuration_T HOUR_CONVERTED_IN_SECONDS
- const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME
- const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME
- const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE
- const Availability_T DEFAULT_CLASS_AVAILABILITY
- const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS
- const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS
- const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR
- const AirlineCode_T DEFAULT_DICO_STUDIED_AIRLINE
- const Yield_T DEFAULT_YIELD_VALUE
- const Yield_T DEFAULT_YIELD_MAX_VALUE

31.5.1 Detailed Description

Handle on the StdAir library context.

Author:

Anh Quan Nguyen <quannaus@users.sourceforge.net>

Date:

20/01/2010 StdAir aims at providing a clean API, and the corresponding C++ implementation, for the basis of Airline IT Business Object Model (BOM), that is, to be used by several other Open Source projects, such as RMOL and OpenTREP.

Install the StdAir library for Airline IT Standard C++ fundaments.

31.5.2 Typedef Documentation

31.5.2.1 typedef date_time_element<0, 23> stdair::hour_t

Type definitions for the date and time elements.

Definition at line 61 of file BasParserHelperTypes.hpp.

31.5.2.2 typedef date_time_element<0, 59> stdair::minute_t

Definition at line 62 of file BasParserHelperTypes.hpp.

31.5.2.3 typedef date_time_element<0, 59> stdair::second_t

Definition at line 63 of file BasParserHelperTypes.hpp.

31.5.2.4 typedef date_time_element<1900, 2100> stdair::year_t

Definition at line 64 of file BasParserHelperTypes.hpp.

31.5.2.5 typedef date_time_element<1, 12> stdair::month_t

Definition at line 65 of file BasParserHelperTypes.hpp.

31.5.2.6 typedef date_time_element<1, 31> stdair::day_t

Definition at line 66 of file BasParserHelperTypes.hpp.

31.5.2.7 typedef std::istreambuf_iterator<char> stdair::base_iterator_t

Definition at line 26 of file BasParserTypes.hpp.

31.5.2.8 typedef boost::spirit::multi_pass<base_iterator_t> stdair::iterator_t

Definition at line 27 of file BasParserTypes.hpp.

31.5.2.9 typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> stdair::int1_p_t

1-digit-integer parser

Definition at line 35 of file BasParserTypes.hpp.

31.5.2.10 typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> stdair::uint2_p_t

2-digit-integer parser

Definition at line 38 of file BasParserTypes.hpp.

31.5.2.11 typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> stdair::uint4_p_t

4-digit-integer parser

Definition at line 41 of file BasParserTypes.hpp.

31.5.2.12 typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> stdair::uint1_4_p_t

Up-to-4-digit-integer parser

Definition at line 44 of file BasParserTypes.hpp.

31.5.2.13 typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> stdair::hour_p_t

Date & time element parsers.

Definition at line 47 of file BasParserTypes.hpp.

31.5.2.14 typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> stdair::minute_p_t

Definition at line 48 of file BasParserTypes.hpp.

31.5.2.15 typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> stdair::second_p_t

Definition at line 49 of file BasParserTypes.hpp.

31.5.2.16 typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> stdair::year_p_t

Definition at line 50 of file BasParserTypes.hpp.

31.5.2.17 typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> stdair::month_p_t

Definition at line 51 of file BasParserTypes.hpp.

31.5.2.18 typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> stdair::day_p_t

Definition at line 52 of file BasParserTypes.hpp.

31.5.2.19 typedef unsigned short stdair::DictionaryKey_T

Dictionary key.

Definition at line 17 of file DictionaryManager.hpp.

31.5.2.20 typedef std::list<AirlineClassList*> stdair::AirlineClassListList_T

Define the segment-features list.

Definition at line 17 of file AirlineClassListTypes.hpp.

31.5.2.21 typedef std::map<const MapKey_T, AirlineClassList*> stdair::AirlineClassListMap_T

Define the segment-features map.

Definition at line 23 of file AirlineClassListTypes.hpp.

31.5.2.22 typedef std::pair<MapKey_T, AirlineClassList*> stdair::AirlineClassListWithKey_T

Define the list of pair<MapKey_T, AirlineCodeList>.

Definition at line 26 of file AirlineClassListTypes.hpp.

31.5.2.23 typedef std::list<AirlineClassListWithKey_T> stdair::AirlineClassListDetailedList_T

Definition at line 27 of file AirlineClassListTypes.hpp.

31.5.2.24 typedef std::list<AirlineFeature*> stdair::AirlineFeatureList_T

Define the airline feature list.

Definition at line 17 of file AirlineFeatureTypes.hpp.

31.5.2.25 typedef std::map<const MapKey_T, AirlineFeature*> stdair::AirlineFeatureMap_T

Define the airline feature map.

Definition at line 23 of file AirlineFeatureTypes.hpp.

31.5.2.26 typedef std::list<AirportPair*> stdair::AirportPairList_T

Define the airport-pair list.

Definition at line 17 of file AirportPairTypes.hpp.

31.5.2.27 typedef std::map<const MapKey_T, AirportPair*> stdair::AirportPairMap_T

Define the airport-pair map.

Definition at line 23 of file AirportPairTypes.hpp.

31.5.2.28 typedef std::pair<MapKey_T, AirportPair*> stdair::AirportPairWithKey_T

Define the list of pair<MapKey_T, AirportPair>.

Definition at line 26 of file AirportPairTypes.hpp.

31.5.2.29 typedef std::list<AirportPairWithKey_T> stdair::AirportPairDetailedList_T

Definition at line 27 of file AirportPairTypes.hpp.

31.5.2.30 typedef std::map<const std::type_info*, BomAbstract*> stdair::HolderMap_T

Definition at line 63 of file BomAbstract.hpp.

31.5.2.31 typedef struct BomID< BookingClass > stdair::BookingClassID_T

Define the booking class ID.

Definition at line 21 of file BomIDTypes.hpp.

31.5.2.32 typedef std::list<BookingClassID_T> stdair::BookingClassIDList_T

Define the list of booking class ID's.

Definition at line 24 of file BomIDTypes.hpp.

31.5.2.33 typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T

Boost Tokeniser.

Definition at line 28 of file BomKeyManager.cpp.

31.5.2.34 typedef std::list<BookingClass*> stdair::BookingClassList_T

Define the booking class list.

Definition at line 17 of file BookingClassTypes.hpp.

31.5.2.35 typedef std::map<const MapKey_T, BookingClass*> stdair::BookingClassMap_T

Define the booking class map.

Definition at line 23 of file BookingClassTypes.hpp.

31.5.2.36 typedef boost::shared_ptr<BookingRequestStruct> stdair::BookingRequestPtr_T

Define the smart pointer to a booking request.

Definition at line 14 of file BookingRequestTypes.hpp.

31.5.2.37 typedef std::string stdair::DemandGeneratorKey_T

Define the hash key for the demand generator.

Definition at line 21 of file BookingRequestTypes.hpp.

31.5.2.38 typedef boost::shared_ptr<BreakPointStruct> stdair::BreakPointPtr_T

Define the smart pointer to a Break Point event .

Definition at line 16 of file BreakPointTypes.hpp.

31.5.2.39 typedef std::list<BreakPointStruct> stdair::BreakPointList_T

Define the list of Break Points.

Definition at line 23 of file BreakPointTypes.hpp.

31.5.2.40 typedef std::list<Bucket*> stdair::BucketList_T

Define the bucket list.

Definition at line 17 of file BucketTypes.hpp.

31.5.2.41 typedef std::map<const MapKey_T, Bucket*> stdair::BucketMap_T

Define the bucket map.

Definition at line 23 of file BucketTypes.hpp.

31.5.2.42 typedef boost::shared_ptr<CancellationStruct> stdair::CancellationPtr_T

Define the smart pointer to a cancellation .

Definition at line 14 of file CancellationTypes.hpp.

31.5.2.43 typedef boost::shared_ptr<ConfigHolderStruct> stdair::ConfigHolderPtr_T

Define the smart pointer to a Config Holder structure.

Definition at line 16 of file ConfigHolderTypes.hpp.

31.5.2.44 typedef std::list<DatePeriod*> stdair::DatePeriodList_T

Define the date-period list.

Definition at line 17 of file DatePeriodTypes.hpp.

31.5.2.45 typedef std::map<const MapKey_T, DatePeriod*> stdair::DatePeriodMap_T

Define the date-period map.

Definition at line 23 of file DatePeriodTypes.hpp.

31.5.2.46 typedef std::pair<MapKey_T, DatePeriod*> stdair::DatePeriodWithKey_T

Define the list of pair<MapKey_T, DatePeriod>.

Definition at line 26 of file DatePeriodTypes.hpp.

31.5.2.47 typedef std::list<DatePeriodWithKey_T> stdair::DatePeriodDetailedList_T

Definition at line 27 of file DatePeriodTypes.hpp.

31.5.2.48 typedef std::pair<const LongDuration_T, EventStruct> stdair::EventListElement_T

Define an element of a event list.

Definition at line 22 of file EventTypes.hpp.

31.5.2.49 typedef std::map<const LongDuration_T, EventStruct> stdair::EventList_T

Define a list of events.

Definition at line 32 of file EventTypes.hpp.

31.5.2.50 typedef std::list<FareFamily*> stdair::FareFamilyList_T

Define the fare family list.

Definition at line 17 of file FareFamilyTypes.hpp.

31.5.2.51 typedef std::map<const MapKey_T, FareFamily*> stdair::FareFamilyMap_T

Define the fare family map.

Definition at line 23 of file FareFamilyTypes.hpp.

31.5.2.52 typedef std::list<FareFeatures*> stdair::FareFeaturesList_T

Define the date-period list.

Definition at line 17 of file FareFeaturesTypes.hpp.

31.5.2.53 typedef std::map<const MapKey_T, FareFeatures*> stdair::FareFeaturesMap_T

Define the date-period map.

Definition at line 23 of file FareFeaturesTypes.hpp.

31.5.2.54 typedef std::pair<MapKey_T, FareFeatures*> stdair::FareFeaturesWithKey_T

Define the list of pair<MapKey_T, FareFeatures>.

Definition at line 26 of file FareFeaturesTypes.hpp.

31.5.2.55 typedef std::list<FareFeaturesWithKey_T> stdair::FareFeaturesDetailedList_T

Definition at line 27 of file FareFeaturesTypes.hpp.

31.5.2.56 typedef std::list<FareOptionStruct> stdair::FareOptionList_T

Define the booking class list.

Definition at line 18 of file FareOptionTypes.hpp.

31.5.2.57 typedef std::map<const std::string, FFDisutilityCurve_T> stdair::FFDisutilityCurveHolder_T

Definition at line 16 of file FFDisutilityCurveHolderStruct.hpp.

31.5.2.58 typedef std::list<FlightDate*> stdair::FlightDateList_T

Define the flight-date list.

Definition at line 17 of file FlightDateTypes.hpp.

31.5.2.59 typedef std::map<const MapKey_T, FlightDate*> stdair::FlightDateMap_T

Define the flight-date map.

Definition at line 24 of file FlightDateTypes.hpp.

31.5.2.60 typedef std::list<FlightPeriod*> stdair::FlightPeriodList_T

Define the flight-period list.

Definition at line 17 of file FlightPeriodTypes.hpp.

31.5.2.61 typedef std::map<const MapKey_T, FlightPeriod*> stdair::FlightPeriodMap_T

Define the flight-period map.

Definition at line 23 of file FlightPeriodTypes.hpp.

31.5.2.62 typedef std::map<const std::string, FRAT5Curve_T> stdair::FRAT5CurveHolder_T

Definition at line 16 of file FRAT5CurveHolderStruct.hpp.

31.5.2.63 typedef std::list<Inventory*> stdair::InventoryList_T

Define the Inventory list.

Definition at line 17 of file InventoryTypes.hpp.

31.5.2.64 typedef std::map<const MapKey_T, Inventory*> stdair::InventoryMap_T

Define the Inventory map.

Definition at line 23 of file InventoryTypes.hpp.

31.5.2.65 typedef std::string stdair::MapKey_T

Key of a STL map.

Definition at line 15 of file key_types.hpp.

31.5.2.66 typedef std::list<std::string> stdair::KeyList_T

List of keys.

Definition at line 18 of file key_types.hpp.

31.5.2.67 typedef std::list<LegCabin*> stdair::LegCabinList_T

Define the leg-cabin list.

Definition at line 17 of file LegCabinTypes.hpp.

31.5.2.68 typedef std::map<const MapKey_T, LegCabin*> stdair::LegCabinMap_T

Define the leg-cabin map.

Definition at line 23 of file LegCabinTypes.hpp.

31.5.2.69 typedef std::list<LegDate*> stdair::LegDateList_T

Define the leg-date list.

Definition at line 17 of file LegDateTypes.hpp.

31.5.2.70 typedef std::map<const MapKey_T, LegDate*> stdair::LegDateMap_T

Define the leg-date map.

Definition at line 23 of file LegDateTypes.hpp.

31.5.2.71 typedef std::list<NestingNode*> stdair::NestingNodeList_T

Define the fare family list.

Definition at line 17 of file NestingNodeTypes.hpp.

31.5.2.72 typedef std::map<const MapKey_T, NestingNode*> stdair::NestingNodeMap_T

Define the fare family map.

Definition at line 23 of file NestingNodeTypes.hpp.

31.5.2.73 typedef std::list<OnDDate*> stdair::OnDDateList_T

Define the O&D date list.

Definition at line 19 of file OnDDateTypes.hpp.

31.5.2.74 typedef std::map<const MapKey_T, OnDDate*> stdair::OnDDateMap_T

Define the OnD date map.

Definition at line 25 of file OnDDateTypes.hpp.

31.5.2.75 typedef std::pair<std::string, YieldDemandPair_T> stdair::StringDemandStructPair_T

Define the yield mean and standard deviation for a certain cabin/class path. This map is mandatory when using the default BOM tree. This map can be empty if yields are charged otherwise (input file, ...)

Definition at line 32 of file OnDDateTypes.hpp.

31.5.2.76 typedef std::map<std::string, YieldDemandPair_T> stdair::StringDemandStructMap_-T

Definition at line 33 of file OnDDateTypes.hpp.

31.5.2.77 typedef std::map<std::string, CabinClassPairList_T> stdair::StringCabinClassPairListMap_T

Define the string matching a (cabin,class) path. (i.e, the string is "Y:M;" for a one leg O&D with the cabin Y and the class M; the string is "Y:M;Y:Y;" for a two legs O&D with the cabin Y and the class M for the first leg, and the cabin Y and the class Y for the second leg).

Definition at line 41 of file OnDDateTypes.hpp.

31.5.2.78 typedef std::pair<std::string, CabinClassPairList_T> stdair::StringCabinClassPair_T

Definition at line 42 of file OnDDateTypes.hpp.

31.5.2.79 typedef std::map<CabinCode_T, WTPDemandPair_T> stdair::CabinForecastMap_T

Define the WTP mean and standard deviation for a certain cabin code. This information is needed to forecast O&D demand per cabin.

Definition at line 48 of file OnDDateTypes.hpp.

31.5.2.80 typedef std::pair<CabinCode_T, WTPDemandPair_T> stdair::CabinForecastPair_T

Definition at line 49 of file OnDDateTypes.hpp.

31.5.2.81 typedef boost:: shared_ptr<OptimisationNotificationStruct> stdair::OptimisationNotificationPtr_T

Define the smart pointer to a optimisation notification.

Definition at line 14 of file OptimisationNotificationTypes.hpp.

31.5.2.82 typedef std::list<Policy*> stdair::PolicyList_T

Define the fare family list.

Definition at line 17 of file PolicyTypes.hpp.

31.5.2.83 typedef std::map<const MapKey_T, Policy*> stdair::PolicyMap_T

Define the fare family map.

Definition at line 23 of file PolicyTypes.hpp.

31.5.2.84 typedef std::list<PosChannel*> stdair::PosChannelList_T

Define the fare-point_of_sale list.

Definition at line 17 of file PosChannelTypes.hpp.

31.5.2.85 typedef std::map<const MapKey_T, PosChannel*> stdair::PosChannelMap_T

Define the fare-point_of_sale map.

Definition at line 23 of file PosChannelTypes.hpp.

31.5.2.86 typedef std::pair<MapKey_T, PosChannel*> stdair::PosChannelWithKey_T

Define the list of pair<MapKey_T, PosChannel>.

Definition at line 26 of file PosChannelTypes.hpp.

31.5.2.87 typedef std::list<PosChannelWithKey_T> stdair::PosChannelDetailedList_T

Definition at line 27 of file PosChannelTypes.hpp.

31.5.2.88 typedef boost::shared_ptr<RMEventStruct> stdair::RMEventPtr_T

Define the smart pointer to a RM event .

Definition at line 16 of file RMEventTypes.hpp.

31.5.2.89 typedef std::list<RMEventStruct> stdair::RMEventList_T

Define the list of RM events.

Definition at line 23 of file RMEventTypes.hpp.

31.5.2.90 typedef std::list<SegmentCabin*> stdair::SegmentCabinList_T

Define the segment-cabin list.

Definition at line 17 of file SegmentCabinTypes.hpp.

31.5.2.91 typedef std::map<const MapKey_T, SegmentCabin*> stdair::SegmentCabinMap_T

Define the segment-cabin map.

Definition at line 23 of file SegmentCabinTypes.hpp.

31.5.2.92 typedef std::list<std::string> stdair::RoutingLegKeyList_T

Definition at line 27 of file SegmentDate.hpp.

31.5.2.93 typedef std::list<SegmentDate*> stdair::SegmentDateList_T

Define the segment-date list.

Definition at line 17 of file SegmentDateTypes.hpp.

31.5.2.94 typedef std::map<const MapKey_T, SegmentDate*> stdair::SegmentDateMap_T

Define the segment-date map.

Definition at line 23 of file SegmentDateTypes.hpp.

31.5.2.95 typedef std::list<SegmentPeriod*> stdair::SegmentPeriodList_T

Define the segment-period list.

Definition at line 17 of file SegmentPeriodTypes.hpp.

31.5.2.96 typedef std::map<const MapKey_T, SegmentPeriod*> stdair::SegmentPeriodMap_T

Define the segment-period map.

Definition at line 23 of file SegmentPeriodTypes.hpp.

31.5.2.97 typedef std::pair<MapKey_T, SegmentPeriod*> stdair::SegmentPeriodWithKey_T

Define the list of pair<MapKey_T, SegmentPeriod>.

Definition at line 26 of file SegmentPeriodTypes.hpp.

31.5.2.98 typedef std::list<SegmentPeriodWithKey_T> stdair::SegmentPeriodDetailedList_T

Definition at line 27 of file SegmentPeriodTypes.hpp.

31.5.2.99 typedef std::list<SegmentSnapshotTable*> stdair::SegmentSnapshotTableList_T

Define the guillotine-block list.

Definition at line 20 of file SegmentSnapshotTableTypes.hpp.

31.5.2.100 typedef std::map<const MapKey_T, SegmentSnapshotTable*> stdair::SegmentSnapshotTableMap_T

Define the guillotine-block map.

Definition at line 27 of file SegmentSnapshotTableTypes.hpp.

31.5.2.101 typedef std::map<const SegmentCabin*, SegmentDataID_T> stdair::SegmentCabinIndexMap_T

Define the map between the segment-cabins and the segment data ID.

Definition at line 30 of file SegmentSnapshotTableTypes.hpp.

31.5.2.102 typedef std::map<const MapKey_T, ClassIndex_T> stdair::ClassIndexMap_T

Define the map between the class and their index.

Definition at line 33 of file SegmentSnapshotTableTypes.hpp.

31.5.2.103 typedef std::list<SimpleNestingStructure*> stdair::SimpleNestingStructureList_T

Define the fare family list.

Definition at line 17 of file SimpleNestingStructureTypes.hpp.

31.5.2.104 typedef std::map<const MapKey_T, SimpleNestingStructure*> stdair::SimpleNestingStructureMap_T

Define the fare family map.

Definition at line 23 of file SimpleNestingStructureTypes.hpp.

31.5.2.105 typedef boost::shared_ptr<SnapshotStruct> stdair::SnapshotPtr_T

Define the smart pointer to a snapshot .

Definition at line 14 of file SnapshotTypes.hpp.

31.5.2.106 typedef std::list<TimePeriod*> stdair::TimePeriodList_T

Define the time-period list.

Definition at line 17 of file TimePeriodTypes.hpp.

31.5.2.107 typedef std::map<const MapKey_T, TimePeriod*> stdair::TimePeriodMap_T

Define the time-period map.

Definition at line 23 of file TimePeriodTypes.hpp.

31.5.2.108 typedef std::pair<MapKey_T, TimePeriod*> stdair::TimePeriodWithKey_T

Define the list of pair<MapKey_T, TimePeriod>.

Definition at line 26 of file TimePeriodTypes.hpp.

31.5.2.109 typedef std::list<TimePeriodWithKey_T> stdair::TimePeriodDetailedList_T

Definition at line 27 of file TimePeriodTypes.hpp.

31.5.2.110 typedef std::list<TravelSolutionStruct> stdair::TravelSolutionList_T

Define the booking class list.

Definition at line 20 of file TravelSolutionTypes.hpp.

31.5.2.111 typedef KeyList_T stdair::SegmentPath_T

Define the segment path key.

Definition at line 26 of file TravelSolutionTypes.hpp.

31.5.2.112 typedef std::list<SegmentPath_T> stdair::SegmentPathList_T

Define the list of segment paths.

Definition at line 29 of file TravelSolutionTypes.hpp.

31.5.2.113 typedef std::map<const ClassCode_T, Availability_T> stdair::ClassAvailabilityMap_T

Define booking class - availability map.

Definition at line 32 of file TravelSolutionTypes.hpp.

31.5.2.114 typedef std::list<ClassAvailabilityMap_T> stdair::ClassAvailabilityMapHolder_T

Define list of booking class - availability maps.

Definition at line 35 of file TravelSolutionTypes.hpp.

31.5.2.115 typedef std::map<const ClassCode_T, BookingClassID_T> stdair::ClassObjectIDMap_T

Define booking class - object ID map.

Definition at line 38 of file TravelSolutionTypes.hpp.

31.5.2.116 typedef std::list<ClassObjectIDMap_T> stdair::ClassObjectIDMapHolder_T

Define list of boking class - object ID maps.

Definition at line 41 of file TravelSolutionTypes.hpp.

31.5.2.117 typedef std::map<const ClassCode_T, YieldValue_T> stdair::ClassYieldMap_T

Define booking class - yield map.

Definition at line 44 of file TravelSolutionTypes.hpp.

31.5.2.118 typedef std::list<ClassYieldMap_T> stdair::ClassYieldMapHolder_T

Define list of booking class - yield maps.

Definition at line 47 of file TravelSolutionTypes.hpp.

31.5.2.119 typedef std::list<BidPriceVector_T> stdair::BidPriceVectorHolder_T

Define list of bid price vectors.

Definition at line 50 of file TravelSolutionTypes.hpp.

31.5.2.120 typedef std::map<const ClassCode_T, const BidPriceVector_T*> stdair::ClassBpvMap_T

Define booking class - bid price reference map.

Definition at line 53 of file TravelSolutionTypes.hpp.

31.5.2.121 typedef std::list<ClassBpvMap_T> stdair::ClassBpvMapHolder_T

Define list of booking class - bid price reference maps.

Definition at line 56 of file TravelSolutionTypes.hpp.

31.5.2.122 typedef std::list<VirtualClassStruct> stdair::VirtualClassList_T

Define the booking class list.

Definition at line 17 of file VirtualClassTypes.hpp.

31.5.2.123 typedef std::map<const YieldLevel_T, VirtualClassStruct> stdair::VirtualClassMap_T

Define the booking class map.

Definition at line 23 of file VirtualClassTypes.hpp.

31.5.2.124 typedef std::list<YieldFeatures*> stdair::YieldFeaturesList_T

Define the date-period list.

Definition at line 17 of file YieldFeaturesTypes.hpp.

31.5.2.125 typedef std::map<const MapKey_T, YieldFeatures*> stdair::YieldFeaturesMap_T

Define the date-period map.

Definition at line 23 of file YieldFeaturesTypes.hpp.

31.5.2.126 typedef std::pair<MapKey_T, YieldFeatures*> stdair::YieldFeaturesWithKey_T

Define the list of pair<MapKey_T, YieldFeatures>.

Definition at line 26 of file YieldFeaturesTypes.hpp.

31.5.2.127 typedef std::list<YieldFeaturesWithKey_T> stdair::YieldFeaturesDetailedList_T

Definition at line 27 of file YieldFeaturesTypes.hpp.

31.5.2.128 typedef std::list<YieldStore*> stdair::YieldStoreList_T

Define the Inventory list.

Definition at line 17 of file YieldStoreTypes.hpp.

31.5.2.129 typedef std::map<const MapKey_T, YieldStore*> stdair::YieldStoreMap_T

Define the Inventory map.

Definition at line 23 of file YieldStoreTypes.hpp.

31.5.2.130 typedef std::string stdair::LocationCode_T

Location code (3-letter-code, e.g., LON).

Definition at line 16 of file stdair_basic_types.hpp.

31.5.2.131 typedef unsigned long int stdair::Distance_T

Define a distance (kilometers).

Definition at line 19 of file stdair_basic_types.hpp.

31.5.2.132 typedef LocationCode_T stdair::AirportCode_T

Define the Airport Code type (3-letter-code, e.g., LHR).

Definition at line 22 of file stdair_basic_types.hpp.

31.5.2.133 typedef LocationCode_T stdair::CityCode_T

City code

Definition at line 25 of file stdair_basic_types.hpp.

31.5.2.134 typedef std::string stdair::KeyDescription_T

Define the key description.

Definition at line 28 of file stdair_basic_types.hpp.

31.5.2.135 typedef std::string stdair::AirlineCode_T

Define the Airline Code type (2-letter-code, e.g., BA).

Definition at line 31 of file stdair_basic_types.hpp.

31.5.2.136 typedef unsigned short stdair::FlightNumber_T

Define the type for flight numbers.

Definition at line 34 of file stdair_basic_types.hpp.

31.5.2.137 typedef unsigned short stdair::TableID_T

Define the type for data table numbers.

Definition at line 37 of file stdair_basic_types.hpp.

31.5.2.138 typedef std::string stdair::CabinCode_T

Define the cabin code (class of service, e.g., first, business, economy).

Definition at line 41 of file stdair_basic_types.hpp.

31.5.2.139 typedef std::string stdair::FamilyCode_T

Define the code of the fare family (e.g., 1, 2, 3, etc.).

Definition at line 44 of file stdair_basic_types.hpp.

31.5.2.140 typedef std::string stdair::PolicyCode_T

Define the code of the policy (e.g., 1, 2, 3, etc.).

Definition at line 47 of file stdair_basic_types.hpp.

31.5.2.141 typedef std::string stdair::NestingStructureCode_T

Define the code of the nesting structure (e.g., "default").

Definition at line 50 of file stdair_basic_types.hpp.

31.5.2.142 typedef std::string stdair::NestingNodeCode_T

Define the code of the nesting node (e.g., 1, 2, 3, etc).

Definition at line 53 of file stdair_basic_types.hpp.

31.5.2.143 typedef std::string stdair::ClassCode_T

Define the booking class code (product segment class, e.g., H, B, K, etc.).

Definition at line 57 of file stdair_basic_types.hpp.

31.5.2.144 typedef unsigned long stdair::Identity_T

Define a identity number.

Definition at line 60 of file stdair_basic_types.hpp.

31.5.2.145 typedef std::string stdair::TripType_T

Type of trip type (RO=outbound of round-trip, RI=inbound of round-trip, OW=one way).

Definition at line 64 of file stdair_basic_types.hpp.

31.5.2.146 typedef double stdair::MonetaryValue_T

Monetary value

Definition at line 67 of file stdair_basic_types.hpp.

31.5.2.147 typedef double stdair::RealNumber_T

Real number

Definition at line 70 of file stdair_basic_types.hpp.

31.5.2.148 typedef double stdair::Percentage_T

Define a percentage value (between 0 and 100%).

Definition at line 73 of file stdair_basic_types.hpp.

31.5.2.149 typedef double stdair::PriceValue_T

Define a price value (e.g., 1000.0 Euros).

Definition at line 76 of file stdair_basic_types.hpp.

31.5.2.150 typedef double stdair::YieldValue_T

Define a yield value (e.g., 1000.0 Euros).

Definition at line 79 of file stdair_basic_types.hpp.

31.5.2.151 typedef std::string stdair::PriceCurrency_T

Define a price currency (e.g., EUR for Euros).

Definition at line 82 of file stdair_basic_types.hpp.

31.5.2.152 typedef double stdair::Revenue_T

Define an amount of revenue.

Define the revenue of a policy

Definition at line 85 of file stdair_basic_types.hpp.

31.5.2.153 typedef double stdair::Multiplier_T

Define the name of a multiplier.

Definition at line 88 of file stdair_basic_types.hpp.

31.5.2.154 typedef double stdair::NbOfSeats_T

Define the number of seats (it can be non integer, because the overbooking can be applied at booking class or PNR level).

Definition at line 92 of file stdair_basic_types.hpp.

31.5.2.155 typedef unsigned int stdair::Count_T

Count

Definition at line 95 of file stdair_basic_types.hpp.

31.5.2.156 typedef short stdair::PartySize_T

Number of passengers (in a group) for a booking.

Definition at line 98 of file stdair_basic_types.hpp.

31.5.2.157 typedef double stdair::NbOfRequests_T

Define a number of requests.

Definition at line 101 of file stdair_basic_types.hpp.

31.5.2.158 typedef NbOfRequests_T stdair::NbOfBookings_T

Define a number of bookings.

Definition at line 104 of file stdair_basic_types.hpp.

31.5.2.159 typedef NbOfRequests_T stdair::NbOfCancellations_T

Define a number of cancellations.

Define a number of cancellations (travellers).

Definition at line 107 of file stdair_basic_types.hpp.

31.5.2.160 typedef unsigned short stdair::NbOfTravelSolutions_T

Define a number of travel solutions (in a travel solution block).

Definition at line 111 of file stdair_basic_types.hpp.

31.5.2.161 typedef std::string stdair::ClassList_String_T

Define the list of class codes as a string.

Definition at line 114 of file stdair_basic_types.hpp.

31.5.2.162 typedef unsigned short stdair::NbOfSegments_T

Define a number of segment-dates (in a path).

Definition at line 117 of file stdair_basic_types.hpp.

31.5.2.163 typedef unsigned short stdair::NbOfAirlines_T

Define a number of airlines (in a path).

Definition at line 120 of file stdair_basic_types.hpp.

31.5.2.164 typedef double stdair::Availability_T

Define an availability.

Definition at line 123 of file stdair_basic_types.hpp.

31.5.2.165 typedef double stdair::Fare_T

Define the price of a travel solution.

Definition at line 126 of file stdair_basic_types.hpp.

31.5.2.166 typedef bool stdair::Flag_T

Define the censorship flag.

Definition at line 129 of file stdair_basic_types.hpp.

31.5.2.167 typedef unsigned int stdair::UnsignedIndex_T

Define the unsigned index type.

Definition at line 132 of file stdair_basic_types.hpp.

31.5.2.168 typedef unsigned int stdair::NbOfClasses_T

Define the number of booking classes.

Definition at line 135 of file stdair_basic_types.hpp.

31.5.2.169 typedef unsigned int stdair::NbOfFareFamilies_T

Define the number of fare families.

Definition at line 138 of file stdair_basic_types.hpp.

31.5.2.170 typedef std::string stdair::Filename_T

File or directory name.

It may contain paths, relative or absolute (e.g., /foo/bar or C:).

Definition at line 144 of file stdair_basic_types.hpp.

31.5.2.171 typedef std::string stdair::FileAddress_T

Define the file address type (e.g. "a_directory/a_filename").

NOTE: That type should be deprecated.

Definition at line 148 of file stdair_basic_types.hpp.

31.5.2.172 typedef float stdair::ProgressPercentage_T

Progress status (usually, a percentage expressed as a floating point number).

Definition at line 152 of file stdair_basic_types.hpp.

31.5.2.173 typedef boost::posix_time::time_duration stdair::Duration_T

Define the type for durations (e.g., elapsed in-flight time).

Definition at line 17 of file stdair_date_time_types.hpp.

31.5.2.174 typedef boost::gregorian::date stdair::Date_T

Define the type for date (e.g., departure date of a flight).

Definition at line 20 of file stdair_date_time_types.hpp.

31.5.2.175 typedef boost::posix_time::time_duration stdair::Time_T

Time

Definition at line 23 of file stdair_date_time_types.hpp.

31.5.2.176 typedef boost::posix_time::ptime stdair::DateTime_T

Define an accurate time (date+time).

Definition at line 26 of file stdair_date_time_types.hpp.

31.5.2.177 typedef boost::gregorian::date_period stdair::DatePeriod_T

Define the Period (e.g., period during which flights depart).

Definition at line 29 of file stdair_date_time_types.hpp.

31.5.2.178 typedef std::string stdair::DOW_String_T

Define the Day-Of-the-Week as a string.

Definition at line 32 of file stdair_date_time_types.hpp.

31.5.2.179 typedef boost::gregorian::date_duration stdair::DateOffset_T

Define the Date Offset (e.g., -1).

Definition at line 35 of file stdair_date_time_types.hpp.

31.5.2.180 typedef int stdair::DayDuration_T

Define a duration in number of days.

Definition at line 38 of file stdair_date_time_types.hpp.

31.5.2.181 typedef bool stdair::SaturdayStay_T

Define the Saturday stay status of a travel.

Define the saturday stay of a tickets.

Definition at line 41 of file stdair_date_time_types.hpp.

31.5.2.182 typedef long int stdair::IntDuration_T

Time duration in (integer) number of seconds

Definition at line 44 of file stdair_date_time_types.hpp.

31.5.2.183 typedef long long int stdair::LongDuration_T

Time duration in (unsigned long long integer) number of milliseconds

Definition at line 47 of file stdair_date_time_types.hpp.

31.5.2.184 typedef float stdair::FloatDuration_T

Duration in (float) number of time units

Definition at line 50 of file stdair_date_time_types.hpp.

31.5.2.185 typedef soci::session stdair::DBSession_T

Database session handler.

Definition at line 20 of file stdair_db.hpp.

31.5.2.186 typedef soci::statement stdair::DBRequestStatement_T

Database request statement handler.

Definition at line 23 of file stdair_db.hpp.

31.5.2.187 typedef std::string stdair::DBConnectionName_T

Define the name of an database connection.

Definition at line 26 of file stdair_db.hpp.

31.5.2.188 typedef bool stdair::ChangeFees_T

Define the availability option allowing the ticket change.

Definition at line 29 of file stdair_demand_types.hpp.

31.5.2.189 typedef bool stdair::NonRefundable_T

Define the refundable availability of a tickets.

Definition at line 32 of file stdair_demand_types.hpp.

31.5.2.190 typedef double stdair::SaturdayStayRatio_T

Define the average ratio (between 0 and 100 percent) of demand with a saturday stay status equal to TRUE.

Definition at line 39 of file stdair_demand_types.hpp.

31.5.2.191 typedef double stdair::ChangeFeesRatio_T

Define the average ratio of demand with change fee availability.

Definition at line 43 of file stdair_demand_types.hpp.

31.5.2.192 typedef double stdair::NonRefundableRatio_T

Define the average ratio of demand with non-refundable availability.

Definition at line 47 of file stdair_demand_types.hpp.

31.5.2.193 typedef double stdair::Disutility_T

Define the disutility of restriction.

Definition at line 50 of file stdair_demand_types.hpp.

31.5.2.194 typedef std::string stdair::PassengerType_T

Define the passenger characteristics, leisure or business for instance (1-letter-code, e.g., L or B).

Definition at line 54 of file stdair_demand_types.hpp.

31.5.2.195 typedef std::string stdair::DistributionPatternId_T

Define the identifier of a distribution pattern (e.g., 1).

Definition at line 57 of file stdair_demand_types.hpp.

31.5.2.196 typedef std::string stdair::CancellationRateCurveId_T

Define the identifier of a cancellation rate curve (e.g., C1).

Definition at line 60 of file stdair_demand_types.hpp.

31.5.2.197 typedef std::string stdair::AirlinePreferenceId_T

Define the identifier of an airline preference set list (e.g., AP1).

Definition at line 63 of file stdair_demand_types.hpp.

31.5.2.198 typedef std::pair<Percentage_T, Percentage_T> stdair::CancellationNoShowRatePair_T

Define a cancellation & and no-show rate pair.

Definition at line 66 of file stdair_demand_types.hpp.

31.5.2.199 typedef std::string stdair::CharacteristicsPatternId_T

Define the identifier of a demand characteristics pattern (e.g. Ch12); for a customer choice model

Definition at line 70 of file stdair_demand_types.hpp.

31.5.2.200 typedef std::string stdair::CharacteristicsIndex_T

Define characteristics component index (e.g. W for WTP)

Definition at line 73 of file stdair_demand_types.hpp.

31.5.2.201 typedef double stdair::WTP_T

Define a Willingness-To-Pay (WTP) (e.g., 1000.0 Euros).

Definition at line 76 of file stdair_demand_types.hpp.

31.5.2.202 typedef boost::tuples::tuple<double, WTP_T> stdair::CharacteristicsWTP_tuple_T

Define the name of a WTP-component of characteristics pattern.

Definition at line 79 of file stdair_demand_types.hpp.

31.5.2.203 typedef std::pair<WTP_T, MeanStdDevPair_T> stdair::WTPDemandPair_T

Define the <WTP, demand> pair type.

Definition at line 82 of file stdair_demand_types.hpp.

31.5.2.204 typedef NbOfRequests_T stdair::NbOfNoShows_T

Define a number of no-shows.

Definition at line 88 of file stdair_demand_types.hpp.

31.5.2.205 typedef double stdair::MatchingIndicator_T

Define a indicator of demand to class matching.

Definition at line 91 of file stdair_demand_types.hpp.

31.5.2.206 typedef std::string stdair::DemandStreamKeyStr_T

Type definition for the hashed key of the DemandStreamKey object.

Definition at line 94 of file stdair_demand_types.hpp.

31.5.2.207 typedef std::string stdair::ChannelLabel_T

Type of booking channel (D=direct, I=indirect, N=oNline, F=oFfline).

Definition at line 97 of file stdair_demand_types.hpp.

31.5.2.208 typedef std::string stdair::FrequentFlyer_T

Type of frequent flyer (P=Platinum, G=Gold, S=Silver, M=Member, N=None).

Definition at line 100 of file stdair_demand_types.hpp.

31.5.2.209 typedef std::string stdair::RequestStatus_T

Define the Request status for booking (1-letter-code, e.g., B: booked, C: cancelled, R: Rejected).

Definition at line 104 of file stdair_demand_types.hpp.

31.5.2.210 typedef std::map<Identity_T, Identity_T> stdair::BookingTSIDMap_T

Define a map between a BookingID and a TravelSolutionID.

Definition at line 107 of file stdair_demand_types.hpp.

31.5.2.211 typedef std::pair<CabinCode_T, ClassCode_T> stdair::CabinClassPair_T

Define a pair (cabin code, class code) e.g., (economy, K).

Definition at line 110 of file stdair_demand_types.hpp.

31.5.2.212 typedef std::list<CabinClassPair_T> stdair::CabinClassPairList_T

Define a list of pair (cabin code, class code).

Definition at line 113 of file stdair_demand_types.hpp.

31.5.2.213 typedef double stdair::ProportionFactor_T

Define the forecast booking requests proportion.

Definition at line 116 of file stdair_demand_types.hpp.

31.5.2.214 typedef std::list<ProportionFactor_T> stdair::ProportionFactorList_T

Define the list of forecast booking requests proportions.

Definition at line 119 of file stdair_demand_types.hpp.

31.5.2.215 typedef std::string stdair::OnDString_T

Define the O&D string key (e.g. "SQ;11,2010-Feb-08;SIN,BKK").

Definition at line 122 of file stdair_demand_types.hpp.

31.5.2.216 typedef std::list<OnDString_T> stdair::OnDStringList_T

Define the list of O&D string key.

Definition at line 125 of file stdair_demand_types.hpp.

31.5.2.217 typedef std::string stdair::EventName_T

Define the name of an event.

Definition at line 14 of file stdair_event_types.hpp.

31.5.2.218 typedef double stdair::NbOfEvents_T

Define a number of events.

Definition at line 17 of file stdair_event_types.hpp.

31.5.2.219 typedef std::string stdair::EventGeneratorKey_T

Define a key string of an event generator.

Definition at line 20 of file stdair_event_types.hpp.

31.5.2.220 typedef double stdair::NbOfFareRules_T

Define a number of fare rules.

Definition at line 12 of file stdair_fare_types.hpp.

31.5.2.221 typedef std::string stdair::NetworkID_T

Define the type for network ID.

Definition at line 23 of file stdair_inventory_types.hpp.

31.5.2.222 typedef std::vector<AirlineCode_T> stdair::AirlineCodeList_T

Define a list of airline code.

Definition at line 26 of file stdair_inventory_types.hpp.

31.5.2.223 typedef std::vector<ClassList_String_T> stdair::ClassList_StringList_T

Define the list of list of class codes as a string.

Definition at line 29 of file stdair_inventory_types.hpp.

31.5.2.224 typedef std::vector<ClassCode_T> stdair::ClassCodeList_T

Define a list of class code.

Definition at line 32 of file stdair_inventory_types.hpp.

31.5.2.225 typedef unsigned short stdair::SubclassCode_T

Define the sub-class code (e.g., 0, 1, 2, etc.). The subclass is a sub-structure for the booking class, allowing to have specific rules for some criteria like POS.

Definition at line 37 of file stdair_inventory_types.hpp.

31.5.2.226 typedef std::string stdair::FlightPathCode_T

Define the flight path code (code made by a suite of flight numbers).

Definition at line 40 of file stdair_inventory_types.hpp.

31.5.2.227 typedef std::map<CabinCode_T, ClassList_String_T> stdair::CabinBookingClassMap_T

Map between the cabin codes and the booking class codes within each cabin.

Definition at line 44 of file stdair_inventory_types.hpp.

31.5.2.228 typedef std::string stdair::CurveKey_T

Curve key for FRAT5 or FF Disutility.

Definition at line 47 of file stdair_inventory_types.hpp.

31.5.2.229 typedef double stdair::CabinCapacity_T

Define the cabin capacity (resource, e.g., 200 seats).

The capacity is expressed as a double to cope with overbooking.

Definition at line 51 of file stdair_inventory_types.hpp.

31.5.2.230 typedef double stdair::NbOfFlightDates_T

Define a number of flight dates.

Definition at line 54 of file stdair_inventory_types.hpp.

31.5.2.231 typedef double stdair::CommittedSpace_T

Define the commited space of a cabin.

Definition at line 57 of file stdair_inventory_types.hpp.

31.5.2.232 typedef double stdair::UPR_T

Define the unsold protection (UPR).

Definition at line 60 of file stdair_inventory_types.hpp.

31.5.2.233 typedef double stdair::BookingLimit_T

Define the value of the booking limit.

Define the Booking Limit.

It is a double, as it allows for overbooking.

Definition at line 63 of file stdair_inventory_types.hpp.

31.5.2.234 typedef double stdair::AuthorizationLevel_T

Define the value of the authorization level.

Definition at line 66 of file stdair_inventory_types.hpp.

31.5.2.235 typedef double stdair::CapacityAdjustment_T

Define the value of the adjustment for cabin capacity.

Definition at line 69 of file stdair_inventory_types.hpp.

31.5.2.236 typedef double stdair::BlockSpace_T

Define the number of seat which could not be used for the booking.

Definition at line 72 of file stdair_inventory_types.hpp.

31.5.2.237 typedef bool stdair::AvailabilityStatus_T

Define an availability status (AVS).

Definition at line 75 of file stdair_inventory_types.hpp.

31.5.2.238 typedef std::vector<Availability_T> stdair::BucketAvailabilities_T

Define a list of availabilities.

Definition at line 78 of file stdair_inventory_types.hpp.

31.5.2.239 typedef double stdair::NbOfYields_T

Define a number of yields.

Definition at line 81 of file stdair_inventory_types.hpp.

31.5.2.240 typedef double stdair::NbOfInventoryControlRules_T

Define a number of InventoryControlRules.

Definition at line 84 of file stdair_inventory_types.hpp.

31.5.2.241 typedef bool stdair::CensorshipFlag_T

Define availibility of booking limit.

Definition at line 87 of file stdair_inventory_types.hpp.

31.5.2.242 typedef short stdair::DTD_T

Define the type of day-to-departure.

Definition at line 90 of file stdair_inventory_types.hpp.

31.5.2.243 typedef short stdair::DCP_T

Define the type of data collection point.

Definition at line 93 of file stdair_inventory_types.hpp.

31.5.2.244 typedef std::list<DCP_T> stdair::DCPList_T

Define the type of data collection point list.

Definition at line 96 of file stdair_inventory_types.hpp.

31.5.2.245 typedef std::map<DTD_T, RealNumber_T> stdair::DTDFratMap_T

Define the DTD (days to departure) frat5 coef map.

Definition at line 99 of file stdair_inventory_types.hpp.

31.5.2.246 typedef std::map<FloatDuration_T, float> stdair::DTDProbMap_T

Define the DTD (days to departure) probability map.

Definition at line 102 of file stdair_inventory_types.hpp.

31.5.2.247 typedef std::vector<CensorshipFlag_T> stdair::CensorshipFlagList_T

Define the list of censorship flags (une list per booking class, one censorship flag per DCP).

Definition at line 106 of file stdair_inventory_types.hpp.

31.5.2.248 typedef double stdair::BookingRatio_T

Define the bookingRatio (for instance OnD bookings over whole class bookings).

Definition at line 110 of file stdair_inventory_types.hpp.

31.5.2.249 typedef double stdair::Yield_T

Define the yield of a virtual class.

Definition at line 113 of file stdair_inventory_types.hpp.

31.5.2.250 typedef unsigned int stdair::YieldLevel_T

Define the yield level (yield as an integer).

Definition at line 116 of file stdair_inventory_types.hpp.

31.5.2.251 typedef std::map<YieldLevel_T, MeanStdDevPair_T> stdair::YieldLevelDemandMap_T

Define the <YieldLevel, demand> demand map.

Definition at line 119 of file stdair_inventory_types.hpp.

31.5.2.252 typedef std::pair<Yield_T, MeanStdDevPair_T> stdair::YieldDemandPair_T

Define the <Yield, demand> pair type.

Definition at line 122 of file stdair_inventory_types.hpp.

31.5.2.253 typedef double stdair::BidPrice_T

Define the Bid-Price.

Definition at line 125 of file stdair_inventory_types.hpp.

31.5.2.254 typedef std::vector<BidPrice_T> stdair::BidPriceVector_T

Define a Bid-Price Vector.

Definition at line 128 of file stdair_inventory_types.hpp.

31.5.2.255 typedef unsigned int stdair::SeatIndex_T

Define the current index of a Bid-Price Vector (for a given LegCabin).

Definition at line 131 of file stdair_inventory_types.hpp.

31.5.2.256 typedef std::string stdair::ControlMode_T

Mode of inventory control.

Definition at line 134 of file stdair_inventory_types.hpp.

31.5.2.257 typedef double stdair::OverbookingRate_T

Define the rate of overbooking

Definition at line 137 of file stdair_inventory_types.hpp.

31.5.2.258 typedef double stdair::ProtectionLevel_T

Define the Protection Level.

It is a double, as it allows for overbooking.

Definition at line 145 of file stdair_inventory_types.hpp.

31.5.2.259 typedef std::vector<double> stdair::EmsrValueList_T

Define the list of EMSR values for the EMSR algorithm.

Definition at line 148 of file stdair_inventory_types.hpp.

31.5.2.260 typedef std::vector<double> stdair::BookingLimitVector_T

Define the vector of booking limits.

It is a vector of double.

Definition at line 152 of file stdair_inventory_types.hpp.

31.5.2.261 typedef std::vector<double> stdair::ProtectionLevelVector_T

Define the vector of protection levels.

It is a vector of double.

Definition at line 156 of file stdair_inventory_types.hpp.

31.5.2.262 typedef boost::multi_array<double, 2> stdair::SnapshotBlock_T

Define a snapshot block.

Definition at line 159 of file stdair_inventory_types.hpp.

31.5.2.263 typedef SnapshotBlock_T::index_range stdair::SnapshotBlockRange_T

Define a range for array view.

Definition at line 162 of file stdair_inventory_types.hpp.

31.5.2.264 typedef SnapshotBlock_T::array_view<1>::type stdair::SegmentCabinDTDSnapshotView_T

Define a view for a given DTD.

Definition at line 165 of file stdair_inventory_types.hpp.

31.5.2.265 typedef SnapshotBlock_T::array_view<2>::type stdair::SegmentCabinDTDRangeSnapshotView_T

Define a view for a given range of DTD.

Definition at line 168 of file stdair_inventory_types.hpp.

31.5.2.266 typedef SnapshotBlock_T::const_array_view<1>::type stdair::ConstSegmentCabinDTDSnapshotView_T

Define a const view for a given DTD.

Definition at line 171 of file stdair_inventory_types.hpp.

31.5.2.267 typedef SnapshotBlock_T::const_array_view<2>::type stdair::ConstSegmentCabinDTDRangeSnapshotView_T

Define a const view for a given range of DTD.

Definition at line 174 of file stdair_inventory_types.hpp.

31.5.2.268 typedef unsigned short stdair::SegmentDataID_T

Define the segment ID within a snapshot data table.

Definition at line 177 of file stdair_inventory_types.hpp.

31.5.2.269 typedef unsigned short stdair::LegDataID_T

Define the leg ID within a snapshot data table.

Definition at line 180 of file stdair_inventory_types.hpp.

31.5.2.270 typedef unsigned short stdair::ClassIndex_T

Define the index type of a class within a snapshot block of a leg/segment.

Definition at line 184 of file stdair_inventory_types.hpp.

31.5.2.271 typedef unsigned int stdair::ReplicationNumber_T

Define the replication number.

Definition at line 24 of file stdair_maths_types.hpp.

31.5.2.272 typedef unsigned long int stdair::ExponentialSeed_T

Define the seed type of an Exponential function.

Definition at line 29 of file stdair_maths_types.hpp.

31.5.2.273 typedef unsigned long int stdair::UniformSeed_T

Define the seed type of an Uniform function.

Definition at line 34 of file stdair_maths_types.hpp.

31.5.2.274 typedef unsigned long int stdair::RandomSeed_T

Seed for the random generation, so that it can be reproductible.

Definition at line 39 of file stdair_maths_types.hpp.

31.5.2.275 typedef boost::minstd_rand stdair::BaseGenerator_T

Random number generator.

Definition at line 44 of file stdair_maths_types.hpp.

31.5.2.276 typedef boost::uniform_real stdair::UniformDistribution_T

Uniform distribution of real numbers (by default, double).

Definition at line 49 of file stdair_maths_types.hpp.

31.5.2.277 typedef boost::variate_generator<BaseGenerator_T&, UniformDistribution_T> stdair::UniformGenerator_T

Uniform random generator.

Definition at line 55 of file stdair_maths_types.hpp.

31.5.2.278 typedef boost::normal_distribution stdair::NormalDistribution_T

Normal distribution of real numbers (by default, double).

Definition at line 60 of file stdair_maths_types.hpp.

31.5.2.279 typedef boost::variate_generator<BaseGenerator_T&, NormalDistribution_T> stdair::NormalGenerator_T

Nornal random generator.

Definition at line 66 of file stdair_maths_types.hpp.

31.5.2.280 typedef boost::exponential_distribution stdair::ExponentialDistribution_T

Type definiton for the exponential distribution (characteristics).

Definition at line 69 of file stdair_maths_types.hpp.

31.5.2.281 typedef boost::variate_generator<BaseGenerator_T&, ExponentialDistribution_T> stdair::ExponentialGenerator_T

Type definition for the exponential distribution random generator.

Definition at line 74 of file stdair_maths_types.hpp.

31.5.2.282 typedef double stdair::MeanValue_T

Define a mean value (e.g., 20.2).

Definition at line 79 of file stdair_maths_types.hpp.

31.5.2.283 typedef double stdair::StdDevValue_T

Define a standard deviation value (e.g., 1.5).

Definition at line 84 of file stdair_maths_types.hpp.

31.5.2.284 typedef std::pair<MeanValue_T, StdDevValue_T> stdair::MeanStdDevPair_T

Define a couple (mean, standart deviation) (e.g., (20.2,1.5)).

Definition at line 89 of file stdair_maths_types.hpp.

31.5.2.285 typedef std::vector<MeanStdDevPair_T> stdair::MeanStdDevPairVector_T

Define a vector of couple (mean, standart deviation)

Definition at line 94 of file stdair_maths_types.hpp.

31.5.2.286 typedef float stdair::Probability_T

Probability.

Definition at line 99 of file stdair_maths_types.hpp.

31.5.2.287 typedef std::string stdair::ForecasterMode_T

Mode of the forecaster.

Definition at line 17 of file stdair_rm_types.hpp.

31.5.2.288 typedef short stdair::HistoricalDataLimit_T

Limit of similar flight-dates used in the forecaster.

Definition at line 24 of file stdair_rm_types.hpp.

31.5.2.289 typedef std::string stdair::OptimizerMode_T

Mode of the forecaster.

Definition at line 27 of file stdair_rm_types.hpp.

31.5.2.290 typedef NbOfBookings_T stdair::PolicyDemand_T

Define the demand for a policy.

Definition at line 30 of file stdair_rm_types.hpp.

31.5.2.291 typedef std::vector<double> stdair::GeneratedDemandVector_T

Define the vector of generated demand (for MC integration use).

It is a vector of double.

Definition at line 34 of file stdair_rm_types.hpp.

31.5.2.292 typedef std::vector<GeneratedDemandVector_T> stdair::GeneratedDemandVectorHolder_T

Define the holder of the generated demand vectors.

Definition at line 37 of file stdair_rm_types.hpp.

31.5.2.293 typedef double stdair::SellupProbability_T

Define the sellup probability.

Definition at line 40 of file stdair_rm_types.hpp.

31.5.2.294 typedef std::vector<NbOfRequests_T> stdair::UncDemVector_T

Define the vector of historical unconstrained demand.

Definition at line 43 of file stdair_rm_types.hpp.

31.5.2.295 typedef std::vector<NbOfBookings_T> stdair::BookingVector_T

Define the vector of historical bookings.

Definition at line 46 of file stdair_rm_types.hpp.

31.5.2.296 typedef double stdair::FRAT5_T

Define the FRAT5 coefficient.

Definition at line 49 of file stdair_rm_types.hpp.

31.5.2.297 typedef std::map<const DTD_T, FRAT5_T> stdair::FRAT5Curve_T

Define the FRAT5 curve.

Definition at line 52 of file stdair_rm_types.hpp.

31.5.2.298 typedef std::map<const DTD_T, double> stdair::FFDisutilityCurve_T

Define the fare family disutility curve.

Definition at line 55 of file stdair_rm_types.hpp.

31.5.2.299 typedef std::map<const DTD_T, double> stdair::SellUpCurve_T

Define the sell-up factor curve.

Definition at line 58 of file stdair_rm_types.hpp.

31.5.2.300 typedef std::map<const DTD_T, double> stdair::DispatchingCurve_T

Define the dispatching factor curve.

Definition at line 61 of file stdair_rm_types.hpp.

31.5.2.301 typedef std::map<BookingClass*, SellUpCurve_T> stdair::BookingClassSellUpCurveMap_T

Define the map between class and sell-up factor curve.

Definition at line 64 of file stdair_rm_types.hpp.

31.5.2.302 typedef std::map<BookingClass*, DispatchingCurve_T> stdair::BookingClassDispatchingCurveMap_T

Define the map between class and dispatching factor curve.

Definition at line 67 of file stdair_rm_types.hpp.

31.5.2.303 typedef std::map<const Yield_T, double> stdair::YieldDemandMap_T

Define the map between the yield of a class and the demand forecast of this class within a policy.

Definition at line 71 of file stdair_rm_types.hpp.

31.5.2.304 typedef unsigned int stdair::NbOfSamples_T

Define the number of samples for the generated demand of booking class

Definition at line 77 of file stdair_rm_types.hpp.

31.5.2.305 typedef boost::shared_ptr<STDAIR_Service> stdair::STDAIR_ServicePtr_T

Pointer on the STDAIR Service handler.

Definition at line 13 of file stdair_service_types.hpp.

31.5.3 Function Documentation

31.5.3.1 const std::string stdair::DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ")

Default value for the BOM tree root key (" -- ROOT -- ").

31.5.3.2 const double stdair::DEFAULT_EPSILON_VALUE (0. *0001*)

Default very small value.

31.5.3.3 const unsigned int stdair::DEFAULT_FLIGHT_SPEED (900)

Default flight speed (number of kilometers per hour).

31.5.3.4 const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES (0. *0*)

Default number of generated flight dates.

31.5.3.5 const Duration_T stdair::NULL_BOOST_TIME_DURATION (- *1*, - *1*, - *1*)

Null time duration (in boost::time_duration unit).

31.5.3.6 const Duration_T stdair::DEFAULT_NULL_DURATION (0, 0, 0)

Default null duration (in boost::time_duration unit).

31.5.3.7 const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)

Default number of days in a year.

31.5.3.8 const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)

Higher value per thousand

31.5.3.9 const DayDuration_T stdair::DEFAULT_DAY_DURATION (0)

Default number of duration days.

31.5.3.10 const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))

Default date period (0-length, i.e., it lasts one day).

31.5.3.11 const DOW_String_T stdair::DEFAULT_DOW_STRING ("0000000")

Default DOW String (e.g., "0000000").

31.5.3.12 const DateOffset_T stdair::DEFAULT_DATE_OFFSET (0)

Default Date Offset (e.g., 0).

31.5.3.13 const Date_T stdair::DEFAULT_DATE (2010, boost::gregorian::Jan, 1)

Default date for the General.

31.5.3.14 const DateTime_T stdair::DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)

Default date-time.

31.5.3.15 const Duration_T stdair::DEFAULT_EPSILON_DURATION (0, 0, 0, 1)

Default epsilon duration (1 nanosecond).

31.5.3.16 const Count_T stdair::SECONDS_IN_ONE_DAY (86400)

Number of seconds in one day.

31.5.3.17 const Count_T stdair::MILLISECONDS_IN_ONE_SECOND (1000)

Number of milliseconds in one second

31.5.3.18 const RandomSeed_T stdair::DEFAULT_RANDOM_SEED (120765987)

Default random seed.

31.5.3.19 const AirportCode_T stdair::AIRPORT_LHR ("LHR")

Default origin airport (e.g., "LHR").

31.5.3.20 const AirportCode_T stdair::AIRPORT_SYD ("SYD")

Default destination airport (e.g., "SYD").

31.5.3.21 const CityCode_T stdair::POS_LHR ("LHR")

London city code (e.g., "LHR").

31.5.3.22 const Date_T stdair::DATE_20110115 (2011, boost::gregorian::Jan, 15)

Date.

31.5.3.23 const Date_T stdair::DATE_20111231 (2011, boost::gregorian::Dec, 31)

31.5.3.24 const DayDuration_T stdair::NO_ADVANCE_PURCHASE (0)

Advance purchase 0 day.

31.5.3.25 const SaturdayStay_T stdair::SATURDAY_STAY (true)

Default saturdayStay value (true).

31.5.3.26 const SaturdayStay_T stdair::NO_SATURDAY_STAY (false)

Default saturdayStay value (false).

31.5.3.27 const ChangeFees_T stdair::CHANGE_FEES (true)

Default change fees value (true).

31.5.3.28 const ChangeFees_T stdair::NO_CHANGE_FEES (false)

Default change fees value (false).

31.5.3.29 const NonRefundable_T stdair::NON_REFUNDABLE (true)

Default non refundable value (true).

31.5.3.30 const NonRefundable_T stdair::NO_NON_REFUNDABLE (false)

Default refundable value (false).

31.5.3.31 const SaturdayStay_T stdair::DEFAULT_BOM_TREE_SATURDAY_STAY (true)

Default saturdayStay value (true).

31.5.3.32 const ChangeFees_T stdair::DEFAULT_BOM_TREE_CHANGE_FEES (true)

Default change fees value (true).

31.5.3.33 const NonRefundable_T stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE (true)

Default non refundable value (true).

31.5.3.34 const DayDuration_T stdair::NO_STAY_DURATION (0)

Stay duration 0 day.

31.5.3.35 const AirlineCode_T stdair::AIRLINE_CODE_BA ("BA")

Airline code "BA".

31.5.3.36 const CabinCode_T stdair::CABIN_Y ("Y")

Cabin 'Y'.

31.5.3.37 const ClassCode_T stdair::CLASS_CODE_Y ("Y")

Class code 'Y'.

31.5.3.38 const ClassCode_T stdair::CLASS_CODE_Q ("Q")

Class code 'Q'.

31.5.3.39 const AirportCode_T stdair::AIRPORT_SIN ("SIN")

Singapour airport (e.g., "SIN").

31.5.3.40 const AirportCode_T stdair::AIRPORT_BKK ("BKK")

Bangkok airport (e.g., "BKK").

31.5.3.41 const CityCode_T stdair::POS_SIN ("SIN")

Singapour city code (e.g., "SIN").

31.5.3.42 const CabinCode_T stdair::CABIN_ECO ("Eco")

Economic cabin (e.g., "Eco").

31.5.3.43 const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ("M")

Frequent flyer tier (e.g., "M" meaning member).

31.5.3.44 const FamilyCode_T stdair::DEFAULT_FAMILY_CODE ("0")

Default family code value ("0").

31.5.3.45 const PolicyCode_T stdair::DEFAULT_POLICY_CODE ("0")

Default policy code value ("0").

31.5.3.46 const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")

Default Nesting Structure Code ("DEFAULT").

31.5.3.47 const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")

Display Nesting Structure Code ("Display Nesting").

31.5.3.48 const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_-CODE ("Yield-Based Nesting")

Display Nesting Structure Code ("Yield-Based Nesting").

31.5.3.49 const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE ("0")

Default Nesting Node Code ("0").

31.5.3.50 const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES (0)

Default number of airlines.

31.5.3.51 const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ("")

Default flight-path code value ("").

31.5.3.52 const Distance_T stdair::DEFAULT_DISTANCE_VALUE (0)

Default distance value (kilometers).

31.5.3.53 const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ("CC")

Default closed class code.

31.5.3.54 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS (0)

Default number of bookings (with counted cancellation) for BookingClass.

31.5.3.55 const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)

Default number of booking (without cancellation) demands for BookingClass.

31.5.3.56 const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)

Default unconstrained demand for BookingClass.

31.5.3.57 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)

Default remaining future demand mean for BookingClass.

31.5.3.58 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_-STANDARD_DEVIATION (0)

Default remaining futre demand standard deviation for BookingClass.

31.5.3.59 const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)

Default number of cancellations for BookingClass.

31.5.3.60 const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS (0)

Default number of no-shows for BookingClass.

31.5.3.61 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY (100. *0*)

Default cabin capacity for Leg cabins.

31.5.3.62 const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE (0. *0*)

Default committed space value for Leg cabins.

31.5.3.63 const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE (0. *0*)

Default commited space value for Leg cabins.

31.5.3.64 const Availability_T stdair::DEFAULT_NULL_AVAILABILITY (0. *0*)

Default null availability (0.0).

31.5.3.65 const Availability_T stdair::DEFAULT_AVAILABILITY (9. *0*)

Default availability (9.0).

31.5.3.66 const Availability_T stdair::MAXIMAL_AVAILABILITY (9999. *0*)

Maximal offered capacity in a cabin.

31.5.3.67 const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG (false)

Default boolean for censorship flag given the status of availability for BookingClass.

31.5.3.68 const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT (9999. *0*)

Default booking limit value for BookingClass.

31.5.3.69 const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999. *0*)

Default authorization level for BookingClass.

31.5.3.70 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_-LEVEL (9999. *0*)

Default MAX value of authorization level for BookingClass.

31.5.3.71 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_-LEVEL (0. *0*)

Default MIN value of authorization level for BookingClass.

31.5.3.72 const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE (0. *0*)

Default over-booking rate for BookingClass.

31.5.3.73 const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE (0. *0*)

Default booking rate for OnD bookings over overall class bookings.

31.5.3.74 const Fare_T stdair::DEFAULT_FARE_VALUE (0. *0*)

Default Fare value.

31.5.3.75 const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE (0. *0*)

Default yield value for a virtual class.

31.5.3.76 const Revenue_T stdair::DEFAULT_REVENUE_VALUE (0. *0*)

Default Revenue value.

31.5.3.77 const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE (100. *0*)

Default load factor value (100%).

31.5.3.78 const Yield_T stdair::DEFAULT_YIELD_VALUE (0. *0*)

Default yield value.

31.5.3.79 const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >:: *max*())

Default yield max value.

31.5.3.80 const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS (0. *0*)

Default number of bookings for YieldRangeStruct_T.

31.5.3.81 const Identity_T stdair::DEFAULT_BOOKING_NUMBER (0)

Default booking number.

31.5.3.82 const NbOfCancellations_T stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS (0. *0*)

Default cancellation number for YieldRangeStruct_T.

31.5.3.83 const NbOfNoShows_T stdair::DEFAULT_YIELD_NB_OF_NOSHOWS (0. *0*)

Default no-shows number for YieldRangeStruct_T.

31.5.3.84 const Availability_T stdair::DEFAULT_YIELD_AVAILABILITY (0. *0*)

Default availability for YieldRangeStruct_T.

31.5.3.85 const CensorshipFlag_T stdair::DEFAULT_YIELD_CENSORSHIPFLAG (false)

Default boolean for booking limit availability for YieldRangeStruct_T.

31.5.3.86 const BookingLimit_T stdair::DEFAULT_YIELD_BOOKING_LIMIT (0. *0*)

Default booking limit value for YieldRangeStruct_T.

31.5.3.87 const OverbookingRate_T stdair::DEFAULT_YIELD_OVERBOOKING_RATE (0. *0*)

Default over-booking rate for YieldRangeStruct_T.

31.5.3.88 const Fare_T stdair::DEFAULT_OND_FARE_VALUE (0. *0*)

Default value of Fare.

31.5.3.89 const Count_T stdair::DEFAULT_PROGRESS_STATUS (0)

Default progress status.

31.5.3.90 const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS (100)

Maximum progress status.

31.5.3.91 const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)

Default reference (oldest) date for the events. No event can occur before that date.

31.5.3.92 const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_DATE, NULL_BOOST_TIME_DURATION)

Default reference (oldest) date-time for the events. No event can occur before that date-time.

31.5.3.93 const PartySize_T stdair::DEFAULT_PARTY_SIZE (1)

Default party size in a request.

31.5.3.94 const DayDuration_T stdair::DEFAULT_STAY_DURATION (7)

Default duration for a stay.

31.5.3.95 const WTP_T stdair::DEFAULT_WTP (1000. *0*)

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

31.5.3.96 const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE)

Default departure date.

31.5.3.97 const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)

Default preferred departure time (08:00).

31.5.3.98 const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE (22)

Default advance purchase.

31.5.3.99 const Date_T stdair::DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_-DEPARTURE_DATE- *DEFAULT_ADVANCE_PURCHASE*)

Default request date.

31.5.3.100 const Duration_T stdair::DEFAULT_REQUEST_TIME (8, 0, 0)

Default preferred departure time (08:00).

31.5.3.101 const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)

Default request date-time.

31.5.3.102 const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN ("M")

Default preferred cabin.

31.5.3.103 const CityCode_T stdair::DEFAULT_POS ("ALL")

Default point-of-sale.

31.5.3.104 const ChannelLabel_T stdair::DEFAULT_CHANNEL ("DC")

Default channel (e.g., "DC" meaning Different Channels).

31.5.3.105 const ChannelLabel_T stdair::CHANNEL_DN ("DN")

DN channel (e.g., direct on-line).

31.5.3.106 const ChannelLabel_T stdair::CHANNEL_IN ("IN")

IN channel (e.g., indirect on-line).

31.5.3.107 const TripType_T stdair::TRIP_TYPE_ONE_WAY ("OW")

Trip type one-way (e.g., "OW").

31.5.3.108 const TripType_T stdair::TRIP_TYPE_ROUND_TRIP ("RT")

Trip type round-trip (e.g., "RT").

31.5.3.109 const TripType_T stdair::TRIP_TYPE_INBOUND ("RI")

Trip type inbound (e.g., "RI").

31.5.3.110 const TripType_T stdair::TRIP_TYPE_OUTBOUND ("RO")

Trip type outbound (e.g., "RO").

31.5.3.111 const FrequentFlyer_T stdair::DEFAULT_FF_TIER ("N")

Default frequent flyer tier (non member).

31.5.3.112 const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME (100. *0*)

Default value of time (expressed as a monetary unit per hour).

31.5.3.113 const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS (3600)

Number of second in one hour

31.5.3.114 const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)

Default Minimal connection time.

31.5.3.115 const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)

Default maximal connection time.

31.5.3.116 const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR (0. *0*)

Default Matching Indicator value.

31.5.3.117 const PriceCurrency_T stdair::DEFAULT_CURRENCY ("EUR")

Default currency (euro).

31.5.3.118 const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS (false)

Default availability status for a travel solution.

31.5.3.119 const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE ("XX")

Default airline code value ("XX").

31.5.3.120 const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE ("")

Default airline code value ("").

31.5.3.121 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER (9999)

Default flight number (9999).

31.5.3.122 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF (255)

Default flight number for fare families (255).

31.5.3.123 const TableID_T stdair::DEFAULT_TABLE_ID (9999)

Default data table number (9999).

31.5.3.124 const Date_T stdair::DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)

Default flight departure date (01/01/1900).

31.5.3.125 const AirportCode_T stdair::DEFAULT_AIRPORT_CODE ("XXX")

Default airport code value ("XXX").

31.5.3.126 const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ("")

Default airport code value (”)..

31.5.3.127 const AirportCode_T stdair::DEFAULT_ORIGIN ("XXX")

Default Origin.

31.5.3.128 const AirportCode_T stdair::DEFAULT_DESTINATION ("YYY")

Default destination.

31.5.3.129 const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")

Default cabin code.

31.5.3.130 const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")

Default fare family Code.

31.5.3.131 const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")

Default null fare family Code ("NoFF").

31.5.3.132 const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")

Default class code value ("X").

31.5.3.133 const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")

Default null class code value ("").

31.5.3.134 const BidPrice_T stdair::DEFAULT_BID_PRICE (0. *0*)

Default Bid-Price.

31.5.3.135 const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)

Maximal number of legs linked to a single flight-date.

Note that the number of derived segments is n∗(n+1)/2 if n is the number of legs.

31.5.3.136 const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)

Maximal number of segments linked to a single O&D (Origin & Destination).

31.5.3.137 const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)

Default seat index (for a bucket and/or Bid-Price Vector slot).

31.5.3.138 const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER (0)

Default number of bookings.

31.5.3.139 const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)

Default capacity adjustment of the cabin.

31.5.3.140 const UPR_T stdair::DEFAULT_NULL_UPR (0)

Default unsold Protection (UPR).

31.5.3.141 const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")

Default value type (within a guillotine block) for fare family.

31.5.3.142 const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")

Default value type (within a guillotine block) for segment-cabin.

31.5.3.143 const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")

Default delimiter for string display (e.g delimiter for inventory key and flight-date key).

31.5.3.144 const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (", ")

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key).

31.5.3.145 const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER (";, ")

Default token for decoding a full string display.

31.5.3.146 template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator∗ (const date_time_element< MIN, MAX > & *o1*, const date_time_element< MIN, MAX > & *o2*) [inline]

Operator∗ overload.

Definition at line 47 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

31.5.3.147 template<int MIN, int MAX> date_time_element<MIN, MAX> stdair::operator+ (const date_time_element< MIN, MAX > & *o1*, const date_time_element< MIN, MAX > & *o2*) [inline]

Operator+ overload.

Definition at line 55 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

31.5.3.148 template void stdair::AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.149 template void stdair::AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.150 template void stdair::BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.151 template void stdair::BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.152 void stdair::intDisplay (std::ostream & *oStream*, const int & *iInt*)

Definition at line 159 of file BookingRequestStruct.cpp.

31.5.3.153 template void stdair::BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.154 template void stdair::BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.155 template void stdair::FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.156 template void stdair::FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.157 template void stdair::FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.158 template void stdair::FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.159 template void stdair::InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.160 template void stdair::InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.161 template void stdair::NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.162 template void stdair::NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.163 template void stdair::NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.164 template void stdair::NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.165 template void stdair::OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.166 template void stdair::OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.167 const boost::char_separator<char> stdair::TokeniserDashSeparator ("-")

Dash delimitor for the tokenisation process.

Referenced by stdair::ParsedKey::getFlightDateKey().

31.5.3.168 const boost::char_separator<char> stdair::TokeniserTimeSeparator (":")

Time delimitor for the tokenisation process.

Referenced by stdair::ParsedKey::getBoardingTime().

31.5.3.169 template void stdair::PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.170 template void stdair::PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.171 template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.172 template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.173 template void stdair::SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.174 template void stdair::SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.175 template void stdair::SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.176 template void stdair::SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.177 template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 > void stdair::serialiseHelper (BOM_OBJECT1 & *ioObject1*, Archive & *ioArchive*, const unsigned int *iFileVersion*) `[inline]`

Definition at line 34 of file CmdBomSerialiser.cpp.

References stdair::BomHolder< BOM >::_bomList, stdair::BomHolder< BOM >::_bomMap, and stdair::FacBomManager::linkWithParent().

31.5.3.178 template void stdair::BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.179 template void stdair::BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.180 template void stdair::Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.181 template void stdair::Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.182 template void stdair::FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.183 template void stdair::FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.184 template void stdair::SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.185 template void stdair::SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.3.186 template void stdair::SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned *int*)

31.5.3.187 template void stdair::SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned *int*)

31.5.4 Variable Documentation

31.5.4.1 const std::string stdair::DOW_STR

Initial value:

```
{"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"}
```

Day names (in English).

Representation of Dow-Of-the-Week

Definition at line 53 of file BasConst.cpp.

Referenced by stdair::DoWStruct::describe().

31.5.4.2 const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD

Default Unconstraining Method (By Expectation-Maximisation).

Default Unconstraining Method (By Time Frame).

Definition at line 140 of file BasConst_Inventory.hpp.

31.5.4.3 const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE

Default Partnership Technique (None).

Definition at line 149 of file BasConst_Inventory.hpp.

31.5.4.4 const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD

Default Forecasting Method (Q Forecasting).

Definition at line 137 of file BasConst_Inventory.hpp.

31.5.4.5 const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD

Default Pre-Optimisation Method (NONE).

Definition at line 143 of file BasConst_Inventory.hpp.

31.5.4.6 const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD

Default Optimisation Method (Leg Based Monte Carlo).

Default Optimisation Method (Leg Based EMSRb).

Definition at line 146 of file BasConst_Inventory.hpp.

31.5.4.7 const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST

Initial value:

```
std::vector<CensorshipFlag_T>()
```

Default list of censorship flag given the status of availability for BookingClass.

Definition at line 253 of file BasConst.cpp.

31.5.4.8 const Date_T stdair::DEFAULT_DICO_STUDIED_DATE

Default DICO studied date.

Definition at line 426 of file BasConst.cpp.

31.5.4.9 const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST

Default airline code list value (empty vector).

Definition at line 436 of file BasConst.cpp.

31.5.4.10 const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST

Default class code list value (empty vector).

Definition at line 478 of file BasConst.cpp.

31.5.4.11 const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()

Default Bid-Price Vector.

Default Bid-Price Vector (empty vector).

Definition at line 484 of file BasConst.cpp.

31.5.4.12 const int stdair::DEFAULT_MAX_DTD = 365

Default value for max day-to-departure (365).

Definition at line 514 of file BasConst.cpp.

Referenced by stdair::SegmentSnapshotTable::initSnapshotBlocks().

31.5.4.13 const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()

Defaut data collection point list.

Definition at line 517 of file BasConst.cpp.

31.5.4.14 const FRAT5Curve_T stdair::FRAT5_CURVE_A

Initial value:

```
DefaultMap::createFRAT5CurveA()
```

FRAT5 curve A for forecasting and optimisation.

FRAT5 curves for forecasting and optimisation.

Definition at line 531 of file BasConst.cpp.

31.5.4.15 const FRAT5Curve_T stdair::FRAT5_CURVE_B

Initial value:

```
DefaultMap::createFRAT5CurveB()
```

FRAT5 curve B for forecasting and optimisation.

Definition at line 545 of file BasConst.cpp.

31.5.4.16 const FRAT5Curve_T stdair::FRAT5_CURVE_C

Initial value:

```
DefaultMap::createFRAT5CurveC()
```

FRAT5 curve C for forecasting and optimisation.

Definition at line 559 of file BasConst.cpp.

31.5.4.17 const FRAT5Curve_T stdair::FRAT5_CURVE_D

Initial value:

```
DefaultMap::createFRAT5CurveD()
```

FRAT5 curve D for forecasting and optimisation.

Definition at line 573 of file BasConst.cpp.

31.5.4.18 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A

Initial value:

```
DefaultMap::createFFDisutilityCurveA()
```

Disutility curve A for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Disutility curves for fare families.

Definition at line 591 of file BasConst.cpp.

31.5.4.19 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B

Initial value:

```
DefaultMap::createFFDisutilityCurveB()
```

Disutility curve B for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 609 of file BasConst.cpp.

31.5.4.20 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C

Initial value:

```
DefaultMap::createFFDisutilityCurveC()
```

Disutility curve C for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 627 of file BasConst.cpp.

31.5.4.21 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D

Initial value:

```
DefaultMap::createFFDisutilityCurveD()
```

Disutility curve D for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 645 of file BasConst.cpp.

31.5.4.22 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E

Initial value:

```
DefaultMap::createFFDisutilityCurveE()
```

Disutility curve E for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 663 of file BasConst.cpp.

31.5.4.23 const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F

Initial value:

```
DefaultMap::createFFDisutilityCurveF()
```

Disutility curve F for forecasting and optimisation. The lower the value (disutility), the higher the demand sells up to higher fare families.

Definition at line 681 of file BasConst.cpp.

31.5.4.24 const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP

Initial value:

```
DefaultDtdFratMap::init()
```

Default frat5 coef map for demand to come forecaster.

Default frat5 coef map.

Definition at line 695 of file BasConst.cpp.

31.5.4.25 const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP

Initial value:

```
DefaultDtdProbMap::init()
```

Default arrival pattern map.

Definition at line 712 of file BasConst.cpp.

31.5.4.26 const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST

Default list of full keys.

Definition at line 736 of file BasConst.cpp.

31.5.4.27 const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY

Array with the indentation spaces needed for all the BOM hierachical levels.

Definition at line 742 of file BasConst.cpp.

31.5.4.28 const std::string stdair::DEFAULT_KEY_FLD_DELIMITER

Default delimiter for string display (e.g delimiter for inventory key and flight-date key). Typically set to ';'.

Referenced by stdair::LegDate::describeRoutingKey(), stdair::SegmentCabin::getFullerKey(), stdair::LegCabin::getFullerKey(), and stdair::ParsedKey::toString().

31.5.4.29 const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER

Default sub delimiter for string display (e.g delimiter for flight number and departure date of a flight-date key). Typically set to ','.

Referenced by stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), stdair::SegmentDateKey::toString(), stdair::PosChannelKey::toString(), stdair::ParsedKey::toString(), stdair::FlightDateKey::toString(), stdair::AirportPairKey::toString(), and stdair::AirlineClassListKey::toString().

31.5.4.30 const boost::char_separator<char> stdair::DEFAULT_KEY_TOKEN_DELIMITER

Default token for decoding a full string display.

Referenced by stdair::BomKeyManager::extractKeys().

31.5.4.31 const Distance_T stdair::DEFAULT_DISTANCE_VALUE

Default distance value, in kilometers (0).

Default distance value (kilometers).

Definition at line 30 of file BasConst_General.hpp.

31.5.4.32 const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE

Default closed class code ("CC").

31.5.4.33 const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS

Default number of bookings (with counted cancellation) for BookingClass (0).

Default number of bookings for BookingClass.

Default number of bookings (0).

Definition at line 27 of file BasConst_General.hpp.

31.5.4.34 const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS

Default number of bookings (without cancellation) for BookingClass (0).

31.5.4.35 const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND

Default unconstrained demand for BookingClass (0).

31.5.4.36 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN

Default remaining future demand mean for BookingClass (0).

31.5.4.37 const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_-STANDARD_DEVIATION

Default remaining futre demand standard deviation for BookingClass (0).

31.5.4.38 const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS

Default number of cancellations for BookingClass (0).

31.5.4.39 const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS

Default number of no-shows for BookingClass (0).

31.5.4.40 const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY

Default cabin capacity for Leg cabins (0.0).

Default cabin capacity for Leg cabins.

Definition at line 21 of file BasConst_General.hpp.

31.5.4.41 const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE

Default commited space value for Leg cabins (0.0).

31.5.4.42 const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE

Default commited space value for Leg cabins (0.0).

31.5.4.43 const Availability_T stdair::DEFAULT_NULL_AVAILABILITY

Default null availability (0.0).

31.5.4.44 const Availability_T stdair::DEFAULT_AVAILABILITY

Default availability (9.0).

31.5.4.45 const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG

Default boolean for censorship flag given the status of availability for BookingClass.

31.5.4.46 const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT

Default booking limit value for BookingClass.

31.5.4.47 const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL

Default authorization level for BookingClass.

31.5.4.48 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_-LEVEL

Default MAX value of authorization level for BookingClass.

31.5.4.49 const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_-LEVEL

Default MIN value of authorization level for BookingClass.

31.5.4.50 const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE

Default over-booking rate for BookingClass.

31.5.4.51 const Fare_T stdair::DEFAULT_FARE_VALUE

Default fare.

Default value of Fare.

Definition at line 36 of file BasConst_General.hpp.

31.5.4.52 const Revenue_T stdair::DEFAULT_REVENUE_VALUE

Default revenue value for BookingClass.

Default revenue value.

Definition at line 42 of file BasConst_General.hpp.

31.5.4.53 const PriceCurrency_T stdair::DEFAULT_CURRENCY

Default currency (euro).

Definition at line 39 of file BasConst_General.hpp.

31.5.4.54 const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE

Default load factor value (100%).

31.5.4.55 const DayDuration_T stdair::DEFAULT_DAY_DURATION

Default number of duration days (0).

Default Duration in days (e.g., 0).

Definition at line 26 of file BasConst_Period_BOM.hpp.

31.5.4.56 const double stdair::DEFAULT_EPSILON_VALUE

Default epsilon value between customer requirements and a fare rule.

Default epsilon value (1e-4).

Definition at line 18 of file BasConst_General.hpp.

31.5.4.57 const AirportCode_T stdair::AIRPORT_LHR

London Heathrow airport (e.g., "LHR").

31.5.4.58 const AirportCode_T stdair::AIRPORT_SYD

Sydney airport (e.g., "SYD").

31.5.4.59 const CityCode_T stdair::POS_LHR

London city code (e.g., "LHR").

31.5.4.60 const DayDuration_T stdair::NO_ADVANCE_PURCHASE

Advance purchase 0 day.

31.5.4.61 const SaturdayStay_T stdair::SATURDAY_STAY

Default saturdayStay value (true).

31.5.4.62 const SaturdayStay_T stdair::NO_SATURDAY_STAY

Default saturdayStay value (false).

31.5.4.63 const ChangeFees_T stdair::CHANGE_FEES

Default change fees value (true).

31.5.4.64 const ChangeFees_T stdair::NO_CHANGE_FEES

Default change fees value (false).

31.5.4.65 const NonRefundable_T stdair::NON_REFUNDABLE

Default non refundable value (true).

31.5.4.66 const NonRefundable_T stdair::NO_NON_REFUNDABLE

Default refundable value (false).

31.5.4.67 const DayDuration_T stdair::NO_STAY_DURATION

Stay duration 0 day.

31.5.4.68 const CabinCode_T stdair::CABIN_Y

Cabin 'Y'.

31.5.4.69 const AirlineCode_T stdair::AIRLINE_CODE_BA

Airline code "BA".

31.5.4.70 const ClassCode_T stdair::CLASS_CODE_Y

Class code 'Y'.

31.5.4.71 const ClassCode_T stdair::CLASS_CODE_Q

Class code 'Q'.

31.5.4.72 const AirportCode_T stdair::AIRPORT_SIN

Singapour airport (e.g., "SIN").

31.5.4.73 const AirportCode_T stdair::AIRPORT_BKK

Bangkok airport (e.g., "BKK").

31.5.4.74 const CityCode_T stdair::POS_SIN

Singapour city code (e.g., "SIN").

31.5.4.75 const CabinCode_T stdair::CABIN_ECO

Economic cabin (e.g., "Eco").

31.5.4.76 const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER

Frequent flyer tier (e.g., "M" meaning member).

31.5.4.77 const Count_T stdair::DEFAULT_PROGRESS_STATUS

Default progress status.

Referenced by stdair::ProgressStatus::reset().

31.5.4.78 const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE

Default reference (oldest) date for the events. No event can occur before that date.

31.5.4.79 const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME

Default reference (oldest) date-time for the events. No event can occur before that date-time.

Referenced by stdair::EventStruct::describe(), stdair::EventStruct::EventStruct(), and stdair::EventStruct::getEventTime().

31.5.4.80 const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS

Maximum progress status.

Referenced by stdair::ProgressStatus::progress().

31.5.4.81 const std::string stdair::DEFAULT_BOM_ROOT_KEY

Default value for the BOM tree root key (" -- ROOT -- ").

31.5.4.82 const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES

Default number of generated flight dates (0).

31.5.4.83 const unsigned int stdair::DEFAULT_FLIGHT_SPEED

Default flight speed (number of kilometers per hour).

31.5.4.84 const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE

Default booking rate for OnD bookings over overall class bookings.

31.5.4.85 const Count_T stdair::SECONDS_IN_ONE_DAY

Number of seconds in one day (86400).

31.5.4.86 const Count_T stdair::MILLISECONDS_IN_ONE_SECOND

Number of milliseconds in one second (1000).

31.5.4.87 const Date_T stdair::DEFAULT_DATE

Default date for the General (1-Jan-2010).

31.5.4.88 const DateTime_T stdair::DEFAULT_DATETIME

Default date-time (1-Jan-2010).

31.5.4.89 const Duration_T stdair::DEFAULT_EPSILON_DURATION

Default epsilon duration (1 nanosecond).

31.5.4.90 const RandomSeed_T stdair::DEFAULT_RANDOM_SEED

Default random seed (120765987).

Referenced by stdair::BookingClass::generateDemandSamples().

31.5.4.91 const Duration_T stdair::NULL_BOOST_TIME_DURATION

Null time duration (in boost::time_duration unit).

Definition at line 23 of file BasConst_TravelSolution.hpp.

31.5.4.92 const Duration_T stdair::DEFAULT_NULL_DURATION

Default null duration (in boost::time_duration unit).

31.5.4.93 const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE

Default value of Availability.

31.5.4.94 const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES

Default number of airlines (0).

31.5.4.95 const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR

Default number of days in a year (365).

31.5.4.96 const ChannelLabel_T stdair::DEFAULT_CHANNEL

Default channel.

Default channel (e.g., direct on-line).

Definition at line 48 of file BasConst_Request.hpp.

31.5.4.97 const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS

Higher value per thousand

Referenced by stdair::DictionaryManager::keyToValue(), and stdair::DictionaryManager::valueToKey().

31.5.4.98 const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE

Default airline code value ("XX").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.99 const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE

Default airline code value ("").

31.5.4.100 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER

Default flight number (9999).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.101 const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF

Default flight number for fare families (255).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.102 const TableID_T stdair::DEFAULT_TABLE_ID

Default data table ID (9999).

31.5.4.103 const Date_T stdair::DEFAULT_DEPARTURE_DATE

Default flight departure date (01/01/1900).

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.104 const AirportCode_T stdair::DEFAULT_AIRPORT_CODE

Default airport code value ("XXX").

31.5.4.105 const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE

Default airport code value ("")..

31.5.4.106 const AirportCode_T stdair::DEFAULT_ORIGIN

Default Origin ("XXX").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.107 const AirportCode_T stdair::DEFAULT_DESTINATION

Default Destination ("XXX").

Referenced by stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.108 const CabinCode_T stdair::DEFAULT_CABIN_CODE

Default Cabin Code ("X").

Referenced by stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveDummySegmentCabin().

31.5.4.109 const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE

Default Fare Family Code ("EcoSaver").

31.5.4.110 const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE

Default null fare family Code ("NoFF").

31.5.4.111 const PolicyCode_T stdair::DEFAULT_POLICY_CODE

Default Policy Code ("0").

31.5.4.112 const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE

Default Nesting Structure Code ("DEFAULT").

31.5.4.113 const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE

Display Nesting Structure Code ("Display Nesting").

31.5.4.114 const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_-CODE

Display Nesting Structure Code ("Yield-Based Nesting").

Referenced by stdair::FacBomManager::resetYieldBasedNestingStructure().

31.5.4.115 const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE

Default Nesting Node Code ("0").

31.5.4.116 const ClassCode_T stdair::DEFAULT_CLASS_CODE

Default class code value ("X").

31.5.4.117 const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE

Default null class code value ("").

31.5.4.118 const BidPrice_T stdair::DEFAULT_BID_PRICE

Default Bid-Price (0.0).

31.5.4.119 const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT

Maximal number of legs linked to a single flight-date (e.g., 7).

Note that the number of derived segments is $n*(n+1)/2$ if n is the number of legs.

31.5.4.120 const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND

Maximal number of segments linked to a single O&D (Origin & Destination)(e.g., 3).

31.5.4.121 const Availability_T stdair::MAXIMAL_AVAILABILITY

Maximal offered capacity in a cabin.

31.5.4.122 const SeatIndex_T stdair::DEFAULT_SEAT_INDEX

Default seat index (for a bucket and/or Bid-Price Vector slot)(e.g., 1).

31.5.4.123 const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER

Default number of bookings.

31.5.4.124 const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT

Default capacity adjustment of the cabin.

31.5.4.125 const UPR_T stdair::DEFAULT_NULL_UPR

Default unsold Protection (UPR).

31.5.4.126 const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE

Default value type (within a guillotine block) for fare family.

31.5.4.127 const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

Default value type (within a guillotine block) for segment-cabin.

31.5.4.128 const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD

Default date period (0-length, i.e., it lasts one day).

31.5.4.129 const DOW_String_T stdair::DEFAULT_DOW_STRING

Default DOW String (e.g., "1111100").

Referenced by stdair::DoWStruct::intersection(), and stdair::DoWStruct::shift().

31.5.4.130 const DateOffset_T stdair::DEFAULT_DATE_OFFSET

Default Date Offset (e.g., 0).

31.5.4.131 const PartySize_T stdair::DEFAULT_PARTY_SIZE

Default party size in a request (e.g., 1).

31.5.4.132 const DayDuration_T stdair::DEFAULT_STAY_DURATION

Default duration for a stay (e.g., 7 days).

31.5.4.133 const WTP_T stdair::DEFAULT_WTP

Default Willingness-to-Pay (WTP, as expressed as a monetary unit).

31.5.4.134 const CityCode_T stdair::DEFAULT_POS

Default Point-Of-Sale (POS, e.g., "WORLD").

31.5.4.135 const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE

Default departure date (e.g., 01-Jan-2011).

31.5.4.136 const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME

Default preferred departure time (e.g., 08:00).

31.5.4.137 const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE

Default advance purchase (e.g., 22 days).

31.5.4.138 const Date_T stdair::DEFAULT_REQUEST_DATE

Default request date (e.g., 10-Jan-2011).

31.5.4.139 const Duration_T stdair::DEFAULT_REQUEST_TIME

Default preferred departure time (e.g., 08:00).

31.5.4.140 const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME

Default request date-time (e.g., 08:00).

31.5.4.141 const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN

Default preferred cabin (e.g., 'M').

31.5.4.142 const ChannelLabel_T stdair::CHANNEL_DN

DN channel (e.g., direct on-line).

31.5.4.143 const ChannelLabel_T stdair::CHANNEL_IN

IN channel (e.g., indirect on-line).

31.5.4.144 const TripType_T stdair::TRIP_TYPE_ONE_WAY

Trip type one-way (e.g., "OW").

Referenced by stdair::BookingRequestStruct::display().

31.5.4.145 const TripType_T stdair::TRIP_TYPE_ROUND_TRIP

Trip type round-trip (e.g., "RT").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

31.5.4.146 const TripType_T stdair::TRIP_TYPE_INBOUND

Trip type inbound (e.g., "RI").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

31.5.4.147 const TripType_T stdair::TRIP_TYPE_OUTBOUND

Trip type outbound (e.g., "RO").

Referenced by stdair::YieldFeatures::isTripTypeValid(), and stdair::FareFeatures::isTripTypeValid().

31.5.4.148 const FrequentFlyer_T stdair::DEFAULT_FF_TIER

Default frequent flyer tier (e.g., non member).

31.5.4.149 const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME

Default value of time (expressed as a monetary unit per hour).

31.5.4.150 const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS

Number of second in one hour

31.5.4.151 const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME

Default Minimal connection time.

31.5.4.152 const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME

Default maximal connection time.

31.5.4.153 const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE

Default flightPathCode value (”).

31.5.4.154 const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY

Default value of Availability.

31.5.4.155 const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS

Default availability status for a travel solution.

31.5.4.156 const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS

Default nember of required seats by the demand.

31.5.4.157 const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR

Default Matching Indicator value between customer requirements and a fare rule.

31.5.4.158 const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE

Default DICO studied airline.

31.5.4.159 const Yield_T stdair::DEFAULT_YIELD_VALUE

Default yield value.

31.5.4.160 const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE

Default yield max value.

31.6 stdair::LOG Namespace Reference

Enumerations

- enum EN_LogLevel {
  CRITICAL = 0, ERROR, NOTIFICATION, WARNING,
  DEBUG, VERBOSE, LAST_VALUE }

Variables

- static const std::string _logLevels [LAST_VALUE]

31.6.1 Detailed Description

Level of logs.

31.6.2 Enumeration Type Documentation

31.6.2.1 enum stdair::LOG::EN_LogLevel

Enumerator:

CRITICAL
ERROR
NOTIFICATION
WARNING
DEBUG
VERBOSE
LAST_VALUE

Definition at line 18 of file stdair_log.hpp.

31.6.3 Variable Documentation

31.6.3.1 const std::string stdair::LOG::_logLevels[LAST_VALUE] [static]

Initial value:

```
{"C", "E", "N", "W", "D", "V"}
```

Definition at line 28 of file stdair_log.hpp.

Referenced by stdair::Logger::log(), stdair::BasLogParams::toShortString(), and stdair::BasLogParams::toString().

31.7 stdair_test Namespace Reference

Classes

- struct BookingClass
- struct Cabin

31.7.1 Detailed Description

Namespace gathering classes and structures for test purposes

31.8 swift Namespace Reference

The wrapper namespace.

Classes

- class SKeymap

  The readline keymap wrapper.

- class SReadline

  The readline library wrapper.

31.8.1 Detailed Description

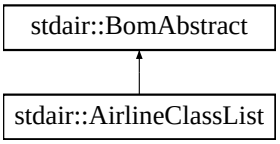
The wrapper namespace. The namespace is also used for other library elements.

32 Class Documentation

32.1 stdair::AirlineClassList Class Reference

Class representing the actual attributes for a segment-features.

`#include <stdair/bom/AirlineClassList.hpp>`Inheritance diagram for stdair::AirlineClassList::



Public Types

- typedef AirlineClassListKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirlineCodeList_T & getAirlineCodeList () const
- const ClassList_StringList_T & getClassCodeList () const
- const HolderMap_T & getHolderMap () const
- const stdair::Yield_T & getYield () const
- const stdair::Fare_T & getFare () const
- void setYield (const Yield_T &iYield)
- void setFare (const Fare_T &iFare)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- AirlineClassList (const Key_T &)
- virtual ~AirlineClassList ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- Yield_T _yield
- Fare_T _fare

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.1.1 Detailed Description

Class representing the actual attributes for a segment-features.

Definition at line 27 of file AirlineClassList.hpp.

32.1.2 Member Typedef Documentation

32.1.2.1 typedef AirlineClassListKey stdair::AirlineClassList::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 38 of file AirlineClassList.hpp.

32.1.3 Constructor & Destructor Documentation

32.1.3.1 stdair::AirlineClassList::AirlineClassList (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 34 of file AirlineClassList.cpp.

32.1.3.2 stdair::AirlineClassList::~AirlineClassList () `[protected, virtual]`

Destructor.

Definition at line 39 of file AirlineClassList.cpp.

32.1.4 Member Function Documentation

32.1.4.1 const Key_T& stdair::AirlineClassList::getKey () const `[inline]`

Get the airline class list key.

Definition at line 44 of file AirlineClassList.hpp.

References _key.

32.1.4.2 BomAbstract* const stdair::AirlineClassList::getParent () const `[inline]`

Get the parent object.

Definition at line 49 of file AirlineClassList.hpp.

References _parent.

32.1.4.3 const AirlineCodeList_T& stdair::AirlineClassList::getAirlineCodeList () const `[inline]`

Get the airline code list (part of the primary key).

Definition at line 54 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::getAirlineCodeList().

32.1.4.4 const ClassList_StringList_T& stdair::AirlineClassList::getClassCodeList () const [inline]

Get the class code list (part of the primary key).

Definition at line 59 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::getClassCodeList().

32.1.4.5 const HolderMap_T& stdair::AirlineClassList::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file AirlineClassList.hpp.

References _holderMap.

32.1.4.6 const stdair::Yield_T& stdair::AirlineClassList::getYield () const [inline]

Get the yield.

Definition at line 69 of file AirlineClassList.hpp.

References _yield.

32.1.4.7 const stdair::Fare_T& stdair::AirlineClassList::getFare () const [inline]

Get the fare.

Definition at line 74 of file AirlineClassList.hpp.

References _fare.

32.1.4.8 void stdair::AirlineClassList::setYield (const Yield_T & *iYield*) [inline]

Definition at line 80 of file AirlineClassList.hpp.

References _yield.

32.1.4.9 void stdair::AirlineClassList::setFare (const Fare_T & *iFare*) [inline]

Definition at line 84 of file AirlineClassList.hpp.

References _fare.

32.1.4.10 void stdair::AirlineClassList::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 95 of file AirlineClassList.hpp.

References toString().

32.1.4.11 void stdair::AirlineClassList::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 104 of file AirlineClassList.hpp.

32.1.4.12 std::string stdair::AirlineClassList::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 43 of file AirlineClassList.cpp.

References _fare, _yield, and describeKey().

Referenced by toStream().

32.1.4.13 const std::string stdair::AirlineClassList::describeKey () const [inline]

Get a string describing the key.

Definition at line 115 of file AirlineClassList.hpp.

References _key, and stdair::AirlineClassListKey::toString().

Referenced by toString().

32.1.4.14 template<class Archive > void stdair::AirlineClassList::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 65 of file AirlineClassList.cpp.

References _fare, _key, and _yield.

32.1.5 Friends And Related Function Documentation

32.1.5.1 friend class FacBom [friend]

Definition at line 28 of file AirlineClassList.hpp.

32.1.5.2 friend class FacCloneBom [friend]

Definition at line 29 of file AirlineClassList.hpp.

32.1.5.3 friend class FacBomManager [friend]

Definition at line 30 of file AirlineClassList.hpp.

32.1.5.4 friend class boost::serialization::access [friend]

Definition at line 31 of file AirlineClassList.hpp.

32.1.6 Member Data Documentation

32.1.6.1 Key_T stdair::AirlineClassList::_key [protected]

Primary key (flight number and departure date).

Definition at line 165 of file AirlineClassList.hpp.

Referenced by describeKey(), getAirlineCodeList(), getClassCodeList(), getKey(), and serialize().

32.1.6.2 BomAbstract* stdair::AirlineClassList::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 170 of file AirlineClassList.hpp.

Referenced by getParent().

32.1.6.3 HolderMap_T stdair::AirlineClassList::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 175 of file AirlineClassList.hpp.

Referenced by getHolderMap().

32.1.6.4 Yield_T stdair::AirlineClassList::_yield [protected]

Definition at line 180 of file AirlineClassList.hpp.

Referenced by getYield(), serialize(), setYield(), and toString().

32.1.6.5 Fare_T stdair::AirlineClassList::_fare [protected]

Definition at line 185 of file AirlineClassList.hpp.

Referenced by getFare(), serialize(), setFare(), and toString().

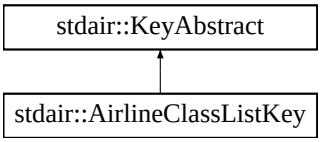
The documentation for this class was generated from the following files:

- stdair/bom/AirlineClassList.hpp
- stdair/bom/AirlineClassList.cpp

32.2 stdair::AirlineClassListKey Struct Reference

Key of airport-pair.

`#include <stdair/bom/AirlineClassListKey.hpp>`Inheritance diagram for stdair::AirlineClassListKey::



Public Member Functions

- AirlineClassListKey (const AirlineCodeList_T &, const ClassList_StringList_T &)
- AirlineClassListKey (const AirlineClassListKey &)
- ~AirlineClassListKey ()
- const AirlineCodeList_T & getAirlineCodeList () const
- const ClassList_StringList_T & getClassCodeList () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.2.1 Detailed Description

Key of airport-pair.

Definition at line 25 of file AirlineClassListKey.hpp.

32.2.2 Constructor & Destructor Documentation

32.2.2.1 stdair::AirlineClassListKey::AirlineClassListKey (const AirlineCodeList_T & *iAirlineCodeList*, const ClassList_StringList_T & *iClassCodeList*)

Constructor.

Definition at line 24 of file AirlineClassListKey.cpp.

32.2.2.2 stdair::AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey & *iKey*)

Copy constructor.

Definition at line 30 of file AirlineClassListKey.cpp.

32.2.2.3 stdair::AirlineClassListKey::~AirlineClassListKey ()

Destructor.

Definition at line 36 of file AirlineClassListKey.cpp.

32.2.3 Member Function Documentation

32.2.3.1 const AirlineCodeList_T& stdair::AirlineClassListKey::getAirlineCodeList () const [inline]

Get the airline code list.

Definition at line 56 of file AirlineClassListKey.hpp.

Referenced by stdair::AirlineClassList::getAirlineCodeList().

32.2.3.2 const ClassList_StringList_T& stdair::AirlineClassListKey::getClassCodeList () const [inline]

Get the class code list.

Definition at line 61 of file AirlineClassListKey.hpp.

Referenced by stdair::AirlineClassList::getClassCodeList().

32.2.3.3 void stdair::AirlineClassListKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file AirlineClassListKey.cpp.

References toString().

32.2.3.4 void stdair::AirlineClassListKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file AirlineClassListKey.cpp.

32.2.3.5 const std::string stdair::AirlineClassListKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file AirlineClassListKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::AirlineClassList::describeKey(), and toStream().

32.2.3.6 template<class Archive > void stdair::AirlineClassListKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 86 of file AirlineClassListKey.cpp.

32.2.4 Friends And Related Function Documentation

32.2.4.1 friend class boost::serialization::access [friend]

Definition at line 26 of file AirlineClassListKey.hpp.

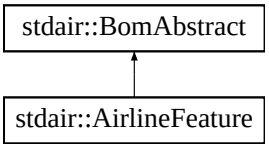
The documentation for this struct was generated from the following files:

- stdair/bom/AirlineClassListKey.hpp
- stdair/bom/AirlineClassListKey.cpp

32.3 stdair::AirlineFeature Class Reference

Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

`#include <stdair/bom/AirlineFeature.hpp>`Inheritance diagram for stdair::AirlineFeature::



Public Types

- typedef AirlineFeatureKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- ForecastingMethod::EN_ForecastingMethod getForecastingMethod () const
- UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod () const
- PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique () const
- PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod () const
- OptimisationMethod::EN_OptimisationMethod getOptimisationMethod () const
- void init (const ForecastingMethod &, const UnconstrainingMethod &, const PreOptimisation-Method &, const OptimisationMethod &, const HistoricalDataLimit_T &, const ControlMode_T &, const PartnershipTechnique &)
- void setForecastingMethod (const ForecastingMethod &iForecastingMethod)
- void setUnconstrainingMethod (const UnconstrainingMethod &iUnconstrainingMethod)
- void setPartnershipTechnique (const PartnershipTechnique &iPartnershipTechnique)
- void setPreOptimisationMethod (const PreOptimisationMethod &iPreOptimisationMethod)
- void setOptimisationMethod (const OptimisationMethod &iOptimisationMethod)

Protected Member Functions

- AirlineFeature (const Key_T &)
- virtual ~AirlineFeature ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- ForecastingMethod _forecastingMethod
- HistoricalDataLimit_T _historicalDataLimit
- ControlMode_T _controlMode
- UnconstrainingMethod _unconstrainingMethod
- PreOptimisationMethod _preOptimisationMethod
- OptimisationMethod _optimisationMethod
- PartnershipTechnique _partnershipTechnique

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.3.1 Detailed Description

Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

Definition at line 25 of file AirlineFeature.hpp.

32.3.2 Member Typedef Documentation

32.3.2.1 typedef AirlineFeatureKey stdair::AirlineFeature::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file AirlineFeature.hpp.

32.3.3 Constructor & Destructor Documentation

32.3.3.1 stdair::AirlineFeature::AirlineFeature (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 14 of file AirlineFeature.cpp.

32.3.3.2 stdair::AirlineFeature::~AirlineFeature () `[protected, virtual]`

Destructor.

Definition at line 34 of file AirlineFeature.cpp.

32.3.4 Member Function Documentation

32.3.4.1 void stdair::AirlineFeature::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 44 of file AirlineFeature.hpp.

References toString().

32.3.4.2 void stdair::AirlineFeature::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 53 of file AirlineFeature.hpp.

32.3.4.3 std::string stdair::AirlineFeature::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 55 of file AirlineFeature.cpp.

References _forecastingMethod, _historicalDataLimit, _optimisationMethod, _partnershipTechnique, _preOptimisationMethod, _unconstrainingMethod, and describeKey().

Referenced by toStream().

32.3.4.4 const std::string stdair::AirlineFeature::describeKey () const [inline]

Get a string describing the key.

Definition at line 64 of file AirlineFeature.hpp.

References _key, and stdair::AirlineFeatureKey::toString().

Referenced by toString().

32.3.4.5 const Key_T& stdair::AirlineFeature::getKey () const [inline]

Get the airline feature primary key (airline code).

Definition at line 73 of file AirlineFeature.hpp.

References _key.

32.3.4.6 BomAbstract* const stdair::AirlineFeature::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 80 of file AirlineFeature.hpp.

References _parent.

32.3.4.7 const HolderMap_T& stdair::AirlineFeature::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 87 of file AirlineFeature.hpp.

References _holderMap.

32.3.4.8 ForecastingMethod::EN_ForecastingMethod stdair::AirlineFeature::getForecastingMethod () const [inline]

Get the forecasting method.

Definition at line 94 of file AirlineFeature.hpp.

References _forecastingMethod, and stdair::ForecastingMethod::getMethod().

Referenced by stdair::Inventory::getForecastingMethod().

32.3.4.9 UnconstrainingMethod::EN_UnconstrainingMethod stdair::AirlineFeature::getUnconstrainingMethod () const [inline]

Get the unconstraining method.

Definition at line 101 of file AirlineFeature.hpp.

References _unconstrainingMethod, and stdair::UnconstrainingMethod::getMethod().

Referenced by stdair::Inventory::getUnconstrainingMethod().

32.3.4.10 PartnershipTechnique::EN_PartnershipTechnique stdair::AirlineFeature::getPartnershipTechnique () const [inline]

Get the partnership technique.

Definition at line 108 of file AirlineFeature.hpp.

References _partnershipTechnique, and stdair::PartnershipTechnique::getTechnique().

Referenced by stdair::Inventory::getPartnershipTechnique().

32.3.4.11 PreOptimisationMethod::EN_PreOptimisationMethod stdair::AirlineFeature::getPreOptimisationMethod () const [inline]

Get the pre-optimisation method.

Definition at line 115 of file AirlineFeature.hpp.

References _preOptimisationMethod, and stdair::PreOptimisationMethod::getMethod().

Referenced by stdair::Inventory::getPreOptimisationMethod().

32.3.4.12 OptimisationMethod::EN_OptimisationMethod stdair::AirlineFeature::getOptimisationMethod () const [inline]

Get the optimisation method.

Definition at line 122 of file AirlineFeature.hpp.

References _optimisationMethod, and stdair::OptimisationMethod::getMethod().

Referenced by stdair::Inventory::getOptimisationMethod().

32.3.4.13 void stdair::AirlineFeature::init (const ForecastingMethod & *iForecastingMethod*, const UnconstrainingMethod & *iUnconstrainingMethod*, const PreOptimisationMethod & *iPreOptimisationMethod*, const OptimisationMethod & *iOptimisationMethod*, const HistoricalDataLimit_T & *iHistoricalDataLimit*, const ControlMode_T & *iControlMode*, const PartnershipTechnique & *iPartnershipTechnique*)

Initialization method.

Parameters:

const ForecastingMethod& Forecasting method.

const UnconstrainingMethod& Unconstraining method.

const PreOptimisationMethod& Pre-optimisation method.

const OptimisationMethodGet& Optimisation method.

const HistoricalDataLimit_T& Historical Data Limit

const ControlMode_T& Control Mode

const PartnershipTechnique& Partnership method.

Definition at line 38 of file AirlineFeature.cpp.

References _controlMode, _forecastingMethod, _historicalDataLimit, _optimisationMethod, _-partnershipTechnique, _preOptimisationMethod, and _unconstrainingMethod.

32.3.4.14 void stdair::AirlineFeature::setForecastingMethod (const ForecastingMethod & *iForecastingMethod*) `[inline]`

Set the forecasting method.

Definition at line 150 of file AirlineFeature.hpp.

References _forecastingMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.3.4.15 void stdair::AirlineFeature::setUnconstrainingMethod (const UnconstrainingMethod & *iUnconstrainingMethod*) `[inline]`

Set the unconstraining method.

Definition at line 157 of file AirlineFeature.hpp.

References _unconstrainingMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.3.4.16 void stdair::AirlineFeature::setPartnershipTechnique (const PartnershipTechnique & *iPartnershipTechnique*) `[inline]`

Set the partnership technique.

Definition at line 164 of file AirlineFeature.hpp.

References _partnershipTechnique.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.3.4.17 void stdair::AirlineFeature::setPreOptimisationMethod (const PreOptimisationMethod & *iPreOptimisationMethod*) [inline]

Set the pre-optimisation method.

Definition at line 171 of file AirlineFeature.hpp.

References _preOptimisationMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.3.4.18 void stdair::AirlineFeature::setOptimisationMethod (const OptimisationMethod & *iOptimisationMethod*) [inline]

Set the optimisation method.

Definition at line 178 of file AirlineFeature.hpp.

References _optimisationMethod.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.3.5 Friends And Related Function Documentation

32.3.5.1 friend class FacBom [friend]

Definition at line 26 of file AirlineFeature.hpp.

32.3.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file AirlineFeature.hpp.

32.3.5.3 friend class FacBomManager [friend]

Definition at line 28 of file AirlineFeature.hpp.

32.3.6 Member Data Documentation

32.3.6.1 Key_T stdair::AirlineFeature::_key [protected]

Primary key (date period).

Definition at line 209 of file AirlineFeature.hpp.

Referenced by describeKey(), and getKey().

32.3.6.2 BomAbstract* stdair::AirlineFeature::_parent [protected]

Pointer on the parent class.

Definition at line 214 of file AirlineFeature.hpp.

Referenced by getParent().

32.3.6.3 HolderMap_T stdair::AirlineFeature::_holderMap [protected]

Map holding the children.

Definition at line 219 of file AirlineFeature.hpp.

Referenced by getHolderMap().

32.3.6.4 ForecastingMethod stdair::AirlineFeature::_forecastingMethod [protected]

The type of forecaster.

Definition at line 224 of file AirlineFeature.hpp.

Referenced by getForecastingMethod(), init(), setForecastingMethod(), and toString().

32.3.6.5 HistoricalDataLimit_T stdair::AirlineFeature::_historicalDataLimit [protected]

The size of the moving average window.

Definition at line 229 of file AirlineFeature.hpp.

Referenced by init(), and toString().

32.3.6.6 ControlMode_T stdair::AirlineFeature::_controlMode [protected]

The type of inventory control.

Definition at line 234 of file AirlineFeature.hpp.

Referenced by init().

32.3.6.7 UnconstrainingMethod stdair::AirlineFeature::_unconstrainingMethod [protected]

The type of unconstraining method.

Definition at line 239 of file AirlineFeature.hpp.

Referenced by getUnconstrainingMethod(), init(), setUnconstrainingMethod(), and toString().

32.3.6.8 PreOptimisationMethod stdair::AirlineFeature::_preOptimisationMethod [protected]

The type of pre-optimisation method.

Definition at line 244 of file AirlineFeature.hpp.

Referenced by getPreOptimisationMethod(), init(), setPreOptimisationMethod(), and toString().

32.3.6.9 OptimisationMethod stdair::AirlineFeature::_optimisationMethod [protected]

The type of optimisation method.

Definition at line 249 of file AirlineFeature.hpp.

Referenced by getOptimisationMethod(), init(), setOptimisationMethod(), and toString().

32.3.6.10 PartnershipTechnique stdair::AirlineFeature::_partnershipTechnique [protected]

The type of partnership technique.

Definition at line 254 of file AirlineFeature.hpp.

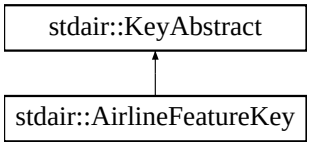
Referenced by getPartnershipTechnique(), init(), setPartnershipTechnique(), and toString().

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

- stdair/bom/AirlineFeature.hpp
- stdair/bom/AirlineFeature.cpp

32.4 stdair::AirlineFeatureKey Struct Reference

`#include <stdair/bom/AirlineFeatureKey.hpp>`Inheritance diagram for stdair::AirlineFeatureKey::



Public Member Functions

- AirlineFeatureKey (const AirlineCode_T &iAirlineCode)
- ~AirlineFeatureKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.4.1 Detailed Description

Key of AirlineFeature.

Definition at line 15 of file AirlineFeatureKey.hpp.

32.4.2 Constructor & Destructor Documentation

32.4.2.1 stdair::AirlineFeatureKey::AirlineFeatureKey (const AirlineCode_T & *iAirlineCode*)

Constructor.

Definition at line 12 of file AirlineFeatureKey.cpp.

32.4.2.2 stdair::AirlineFeatureKey::~AirlineFeatureKey ()

Destructor.

Definition at line 17 of file AirlineFeatureKey.cpp.

32.4.3 Member Function Documentation

32.4.3.1 const AirlineCode_T& stdair::AirlineFeatureKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 27 of file AirlineFeatureKey.hpp.

32.4.3.2 void stdair::AirlineFeatureKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 21 of file AirlineFeatureKey.cpp.

References toString().

32.4.3.3 void stdair::AirlineFeatureKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 26 of file AirlineFeatureKey.cpp.

32.4.3.4 const std::string stdair::AirlineFeatureKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file AirlineFeatureKey.cpp.

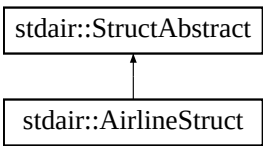
Referenced by stdair::AirlineFeature::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/AirlineFeatureKey.hpp
- stdair/bom/AirlineFeatureKey.cpp

32.5 stdair::AirlineStruct Struct Reference

`#include <stdair/bom/AirlineStruct.hpp>`Inheritance diagram for stdair::AirlineStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const std::string & getAirlineName () const
- void setAirlineCode (const AirlineCode_T &iAirlineCode)
- void setAirlineName (const std::string &iAirlineName)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- AirlineStruct (const AirlineCode_T &, const std::string &iAirlineName)
- AirlineStruct ()
- AirlineStruct (const AirlineStruct &)
- ~AirlineStruct ()

32.5.1 Detailed Description

Structure holding parameters describing an airline.

Definition at line 18 of file AirlineStruct.hpp.

32.5.2 Constructor & Destructor Documentation

32.5.2.1 stdair::AirlineStruct::AirlineStruct (const AirlineCode_T & *iAirlineCode*, const std::string & *iAirlineName*)

Main constructor.

Definition at line 24 of file AirlineStruct.cpp.

32.5.2.2 stdair::AirlineStruct::AirlineStruct ()

Default constructor.

Definition at line 15 of file AirlineStruct.cpp.

32.5.2.3 stdair::AirlineStruct::AirlineStruct (const AirlineStruct & *iAirlineStruct*)

Default copy constructor.

Definition at line 19 of file AirlineStruct.cpp.

32.5.2.4 stdair::AirlineStruct::~AirlineStruct ()

Destructor.

Definition at line 30 of file AirlineStruct.cpp.

32.5.3 Member Function Documentation

32.5.3.1 const AirlineCode_T& stdair::AirlineStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 22 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::to_base(), and stdair::DBManagerForAirlines::updateAirlineInDB().

32.5.3.2 const std::string& stdair::AirlineStruct::getAirlineName () const [inline]

Get the airline name.

Definition at line 27 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::to_base(), and stdair::DBManagerForAirlines::updateAirlineInDB().

32.5.3.3 void stdair::AirlineStruct::setAirlineCode (const AirlineCode_T & *iAirlineCode*) [inline]

Set the airline code.

Definition at line 33 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::from_base().

32.5.3.4 void stdair::AirlineStruct::setAirlineName (const std::string & *iAirlineName*) [inline]

Set the airline name.

Definition at line 38 of file AirlineStruct.hpp.

Referenced by soci::type_conversion< stdair::AirlineStruct >::from_base().

32.5.3.5 void stdair::AirlineStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 34 of file AirlineStruct.cpp.

References describe().

32.5.3.6 void stdair::AirlineStruct::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 39 of file AirlineStruct.cpp.

32.5.3.7 const std::string stdair::AirlineStruct::describe () const `[virtual]`

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 43 of file AirlineStruct.cpp.

Referenced by toStream().

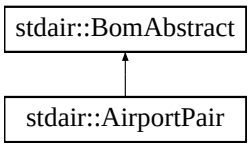
The documentation for this struct was generated from the following files:

- stdair/bom/AirlineStruct.hpp
- stdair/bom/AirlineStruct.cpp

32.6 stdair::AirportPair Class Reference

Class representing the actual attributes for an airport-pair.

`#include <stdair/bom/AirportPair.hpp>`Inheritance diagram for stdair::AirportPair::



Public Types

- typedef AirportPairKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const

Protected Member Functions

- AirportPair (const Key_T &)
- virtual ~AirportPair ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.6.1 Detailed Description

Class representing the actual attributes for an airport-pair.

Definition at line 18 of file AirportPair.hpp.

32.6.2 Member Typedef Documentation

32.6.2.1 typedef AirportPairKey stdair::AirportPair::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file AirportPair.hpp.

32.6.3 Constructor & Destructor Documentation

32.6.3.1 stdair::AirportPair::AirportPair (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 27 of file AirportPair.cpp.

32.6.3.2 stdair::AirportPair::~AirportPair () `[protected, virtual]`

Destructor.

Definition at line 32 of file AirportPair.cpp.

32.6.4 Member Function Documentation

32.6.4.1 void stdair::AirportPair::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file AirportPair.hpp.

References toString().

32.6.4.2 void stdair::AirportPair::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file AirportPair.hpp.

32.6.4.3 std::string stdair::AirportPair::toString () const `[virtual]`

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file AirportPair.cpp.

References describeKey().

Referenced by toStream().

32.6.4.4 const std::string stdair::AirportPair::describeKey () const `[inline]`

Get a string describing the key.

Definition at line 57 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::toString().

Referenced by toString().

32.6.4.5 const Key_T& stdair::AirportPair::getKey () const `[inline]`

Get the primary key (origin airport, destination airport).

Definition at line 66 of file AirportPair.hpp.

References _key.

32.6.4.6 const AirportCode_T& stdair::AirportPair::getBoardingPoint () const `[inline]`

Get the origin airport.

Definition at line 73 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::getBoardingPoint().

32.6.4.7 const AirportCode_T& stdair::AirportPair::getOffPoint () const [inline]

Get the destination airport.

Definition at line 80 of file AirportPair.hpp.

References _key, and stdair::AirportPairKey::getOffPoint().

32.6.4.8 BomAbstract* const stdair::AirportPair::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 87 of file AirportPair.hpp.

References _parent.

32.6.4.9 const HolderMap_T& stdair::AirportPair::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 94 of file AirportPair.hpp.

References _holderMap.

32.6.5 Friends And Related Function Documentation

32.6.5.1 friend class FacBom [friend]

Definition at line 19 of file AirportPair.hpp.

32.6.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file AirportPair.hpp.

32.6.5.3 friend class FacBomManager [friend]

Definition at line 21 of file AirportPair.hpp.

32.6.6 Member Data Documentation

32.6.6.1 Key_T stdair::AirportPair::_key [protected]

Primary key (flight number and departure date).

Definition at line 124 of file AirportPair.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), and getOffPoint().

32.6.6.2 BomAbstract* stdair::AirportPair::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 129 of file AirportPair.hpp.

Referenced by getParent().

32.6.6.3 HolderMap_T stdair::AirportPair::_holderMap [protected]

Map holding the children.

Definition at line 134 of file AirportPair.hpp.

Referenced by getHolderMap().

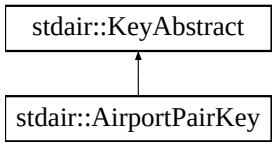
The documentation for this class was generated from the following files:

- stdair/bom/AirportPair.hpp
- stdair/bom/AirportPair.cpp

32.7 stdair::AirportPairKey Struct Reference

Key of airport-pair.

`#include <stdair/bom/AirportPairKey.hpp>`Inheritance diagram for stdair::AirportPairKey::



Public Member Functions

- AirportPairKey (const stdair::AirportCode_T &, const stdair::AirportCode_T &)
- AirportPairKey (const AirportPairKey &)
- ~AirportPairKey ()
- const stdair::AirportCode_T & getBoardingPoint () const
- const stdair::AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.7.1 Detailed Description

Key of airport-pair.

Definition at line 16 of file AirportPairKey.hpp.

32.7.2 Constructor & Destructor Documentation

32.7.2.1 stdair::AirportPairKey::AirportPairKey (const stdair::AirportCode_T & *iBoardingPoint*, const stdair::AirportCode_T & *iOffPoint*)

Main constructor.

Definition at line 22 of file AirportPairKey.cpp.

32.7.2.2 stdair::AirportPairKey::AirportPairKey (const AirportPairKey & *iKey*)

Copy constructor.

Definition at line 28 of file AirportPairKey.cpp.

32.7.2.3 stdair::AirportPairKey::~AirportPairKey ()

Destructor.

Definition at line 34 of file AirportPairKey.cpp.

32.7.3 Member Function Documentation

32.7.3.1 const stdair::AirportCode_T& stdair::AirportPairKey::getBoardingPoint () const `[inline]`

Get the boarding point.

Definition at line 36 of file AirportPairKey.hpp.

Referenced by stdair::AirportPair::getBoardingPoint().

32.7.3.2 const stdair::AirportCode_T& stdair::AirportPairKey::getOffPoint () const `[inline]`

Get the arrival point.

Definition at line 43 of file AirportPairKey.hpp.

Referenced by stdair::AirportPair::getOffPoint().

32.7.3.3 void stdair::AirportPairKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file AirportPairKey.cpp.

References toString().

32.7.3.4 void stdair::AirportPairKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file AirportPairKey.cpp.

32.7.3.5 const std::string stdair::AirportPairKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 47 of file AirportPairKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::AirportPair::describeKey(), stdair::BomRetriever::retrieveAirportPairFromKeySet(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/AirportPairKey.hpp
- stdair/bom/AirportPairKey.cpp

32.8 stdair::BasChronometer Struct Reference

```
#include <stdair/basic/BasChronometer.hpp>
```

Public Member Functions

- BasChronometer ()
- void start ()
- std::string getStart () const
- double elapsed () const

32.8.1 Detailed Description

Structure allowing measuring the time elapsed between two events.

Definition at line 14 of file BasChronometer.hpp.

32.8.2 Constructor & Destructor Documentation

32.8.2.1 stdair::BasChronometer::BasChronometer ()

Constructor.

Definition at line 12 of file BasChronometer.cpp.

32.8.3 Member Function Documentation

32.8.3.1 void stdair::BasChronometer::start ()

Start the chronometer from the local time

The elapsed time given is the one elapsed since the start is launched.

Definition at line 16 of file BasChronometer.cpp.

32.8.3.2 std::string stdair::BasChronometer::getStart () const [inline]

Get the start time.

Definition at line 24 of file BasChronometer.hpp.

32.8.3.3 double stdair::BasChronometer::elapsed () const

Return the time elapsed since the structure has been instanciated.

That elapsed time is expressed in seconds.

Definition at line 26 of file BasChronometer.cpp.

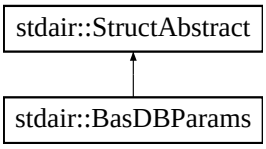
The documentation for this struct was generated from the following files:

- stdair/basic/BasChronometer.hpp
- stdair/basic/BasChronometer.cpp

32.9 stdair::BasDBParams Struct Reference

Structure holding the parameters for connection to a database.

`#include <stdair/basic/BasDBParams.hpp>`Inheritance diagram for stdair::BasDBParams::



Public Member Functions

- const std::string & getUser () const
- const std::string & getPassword () const
- const std::string & getHost () const
- const std::string & getPort () const
- const std::string & getDBName () const
- void setUser (const std::string &iUser)
- void setPassword (const std::string &iPasswd)
- void setHost (const std::string &iHost)
- void setPort (const std::string &iPort)
- void setDBName (const std::string &iDBName)

- bool check () const
- const std::string describe () const
- std::string toShortString () const
- std::string toString () const
- BasDBParams (const std::string &iDBUser, const std::string &iDBPasswd, const std::string &iDB-Host, const std::string &iDBPort, const std::string &iDBName)
- BasDBParams ()
- BasDBParams (const BasDBParams &)
- ~BasDBParams ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

32.9.1 Detailed Description

Structure holding the parameters for connection to a database.

Definition at line 19 of file BasDBParams.hpp.

32.9.2 Constructor & Destructor Documentation

32.9.2.1 stdair::BasDBParams::BasDBParams (const std::string & *iDBUser*, const std::string & *iDBPasswd*, const std::string & *iDBHost*, const std::string & *iDBPort*, const std::string & *iDBName*)

Main Constructor.

Definition at line 24 of file BasDBParams.cpp.

32.9.2.2 stdair::BasDBParams::BasDBParams ()

Default Constructor.

Definition at line 13 of file BasDBParams.cpp.

32.9.2.3 stdair::BasDBParams::BasDBParams (const BasDBParams & *iDBParams*)

Default copy constructor.

Definition at line 17 of file BasDBParams.cpp.

32.9.2.4 stdair::BasDBParams::~BasDBParams ()

Destructor.

Definition at line 34 of file BasDBParams.cpp.

32.9.3 Member Function Documentation

32.9.3.1 const std::string& stdair::BasDBParams::getUser () const `[inline]`

Get the database user name.

Definition at line 23 of file BasDBParams.hpp.

32.9.3.2 const std::string& stdair::BasDBParams::getPassword () const [inline]

Get the database user password.

Definition at line 28 of file BasDBParams.hpp.

32.9.3.3 const std::string& stdair::BasDBParams::getHost () const [inline]

Get the database host name.

Definition at line 33 of file BasDBParams.hpp.

32.9.3.4 const std::string& stdair::BasDBParams::getPort () const [inline]

Get the database port number.

Definition at line 38 of file BasDBParams.hpp.

32.9.3.5 const std::string& stdair::BasDBParams::getDBName () const [inline]

Get the database name.

Definition at line 43 of file BasDBParams.hpp.

32.9.3.6 void stdair::BasDBParams::setUser (const std::string & *iUser*) [inline]

Set the database user name.

Definition at line 50 of file BasDBParams.hpp.

32.9.3.7 void stdair::BasDBParams::setPassword (const std::string & *iPasswd*) [inline]

Set the database password.

Definition at line 55 of file BasDBParams.hpp.

32.9.3.8 void stdair::BasDBParams::setHost (const std::string & *iHost*) [inline]

Set the database host name.

Definition at line 60 of file BasDBParams.hpp.

32.9.3.9 void stdair::BasDBParams::setPort (const std::string & *iPort*) [inline]

Set the database port number.

Definition at line 65 of file BasDBParams.hpp.

32.9.3.10 void stdair::BasDBParams::setDBName (const std::string & *iDBName*) [inline]

Set the database name.

Definition at line 70 of file BasDBParams.hpp.

32.9.3.11 bool stdair::BasDBParams::check () const

Check that all the parameters are fine.

Definition at line 57 of file BasDBParams.cpp.

32.9.3.12 const std::string stdair::BasDBParams::describe () const [virtual]

Get the serialised version of the DBParams structure.

Implements stdair::StructAbstract.

Definition at line 38 of file BasDBParams.cpp.

References toString().

32.9.3.13 std::string stdair::BasDBParams::toShortString () const

Get a short display of the DBParams structure.

Definition at line 43 of file BasDBParams.cpp.

32.9.3.14 std::string stdair::BasDBParams::toString () const

Get the serialised version of the DBParams structure.

Definition at line 50 of file BasDBParams.cpp.

Referenced by describe().

32.9.3.15 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.9.3.16 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/BasDBParams.hpp
- stdair/basic/BasDBParams.cpp

32.10 stdair::BasFileMgr Struct Reference

```
#include <stdair/basic/BasFileMgr.hpp>
```

Static Public Member Functions

- static bool doesExistAndIsReadable (const std::string &iFilepath)

32.10.1 Detailed Description

Helper class for operations on files and on the file-system.

Definition at line 13 of file BasFileMgr.hpp.

32.10.2 Member Function Documentation

32.10.2.1 bool stdair::BasFileMgr::doesExistAndIsReadable (const std::string & *iFilepath*) [static]

Definition at line 23 of file BasFileMgr.cpp.

Referenced by stdair::BomINIImport::importINIConfig().

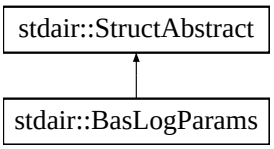
The documentation for this struct was generated from the following files:

- stdair/basic/BasFileMgr.hpp
- stdair/basic/BasFileMgr.cpp

32.11 stdair::BasLogParams Struct Reference

Structure holding parameters for logging.

`#include <stdair/basic/BasLogParams.hpp>`Inheritance diagram for stdair::BasLogParams::



Public Member Functions

- const LOG::EN_LogLevel & getLogLevel () const
- std::ostream & getLogStream () const
- const bool getForcedInitialisationFlag () const
- void setForcedInitialisationFlag (const bool iForceMultipleInstance)
- bool check () const
- const std::string describe () const
- std::string toShortString () const
- std::string toString () const
- BasLogParams (const LOG::EN_LogLevel iLogLevel, std::ostream &ioLogOutputStream, const bool iForceMultipleInstance=false)
- BasLogParams (const BasLogParams &)
- ~BasLogParams ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Friends

- class Logger

32.11.1 Detailed Description

Structure holding parameters for logging.

Definition at line 19 of file BasLogParams.hpp.

32.11.2 Constructor & Destructor Documentation

32.11.2.1 stdair::BasLogParams::BasLogParams (const LOG::EN_LogLevel *iLogLevel*, std::ostream & *ioLogOutputStream*, const bool *iForceMultipleInstance* = `false`)

Main Constructor.

Parameters:

← ***const*** LOG::EN_LogLevel Level of the log (e.g., DEBUG)

↔ ***std::ostream&*** (STL) Stream to log into.

← ***const*** bool Whether or not multiple initialisation should be forced.

Definition at line 27 of file BasLogParams.cpp.

32.11.2.2 stdair::BasLogParams::BasLogParams (const BasLogParams & *iLogParams*)

Copy constructor.

Definition at line 21 of file BasLogParams.cpp.

32.11.2.3 stdair::BasLogParams::~BasLogParams ()

Destructor.

Definition at line 35 of file BasLogParams.cpp.

32.11.3 Member Function Documentation

32.11.3.1 const LOG::EN_LogLevel& stdair::BasLogParams::getLogLevel () const `[inline]`

Get the log level.

Definition at line 26 of file BasLogParams.hpp.

32.11.3.2 std::ostream& stdair::BasLogParams::getLogStream () const `[inline]`

Get the log output stream.

Definition at line 33 of file BasLogParams.hpp.

32.11.3.3 const bool stdair::BasLogParams::getForcedInitialisationFlag () const `[inline]`

State whether or not multiple initialisations are to be forced.

Definition at line 40 of file BasLogParams.hpp.

32.11.3.4 void stdair::BasLogParams::setForcedInitialisationFlag (const bool *iForceMultipleInstance*) `[inline]`

State whether or not multiple initialisations are to be forced.

Definition at line 49 of file BasLogParams.hpp.

32.11.3.5 bool stdair::BasLogParams::check () const

Check that all the parameters are fine.

32.11.3.6 const std::string stdair::BasLogParams::describe () const `[virtual]`

Get the serialised version of the DBParams structure.

Implements stdair::StructAbstract.

Definition at line 39 of file BasLogParams.cpp.

References toString().

32.11.3.7 std::string stdair::BasLogParams::toShortString () const

Get a short display of the LOGParams structure.

Definition at line 44 of file BasLogParams.cpp.

References stdair::LOG::_logLevels.

32.11.3.8 std::string stdair::BasLogParams::toString () const

Get the serialised version of the LOGParams structure.

Definition at line 52 of file BasLogParams.cpp.

References stdair::LOG::_logLevels.

Referenced by describe().

32.11.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.11.3.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

32.11.4 Friends And Related Function Documentation

32.11.4.1 friend class Logger [friend]

Definition at line 20 of file BasLogParams.hpp.

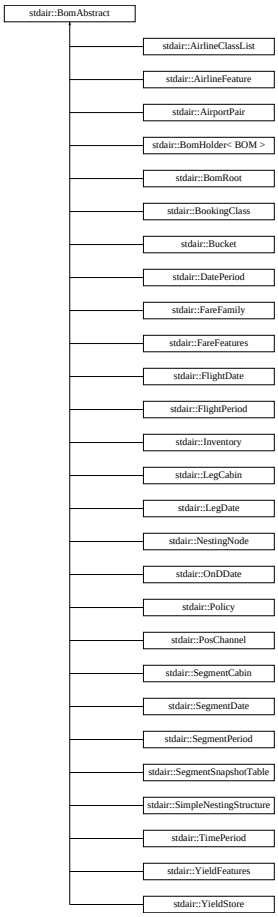
The documentation for this struct was generated from the following files:

- stdair/basic/BasLogParams.hpp
- stdair/basic/BasLogParams.cpp

32.12 stdair::BomAbstract Class Reference

Base class for the Business Object Model (BOM) layer.

`#include <stdair/bom/BomAbstract.hpp>`Inheritance diagram for stdair::BomAbstract::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const =0
- virtual void fromStream (std::istream &ioIn)=0

- virtual std::string toString () const =0
- virtual ~BomAbstract ()

Protected Member Functions

- BomAbstract ()
- BomAbstract (const BomAbstract &)

32.12.1 Detailed Description

Base class for the Business Object Model (BOM) layer.

Definition at line 24 of file BomAbstract.hpp.

32.12.2 Constructor & Destructor Documentation

32.12.2.1 stdair::BomAbstract::BomAbstract () `[inline, protected]`

Protected Default Constructor to ensure this class is abtract.

Definition at line 53 of file BomAbstract.hpp.

32.12.2.2 stdair::BomAbstract::BomAbstract (const BomAbstract &) `[inline, protected]`

Definition at line 54 of file BomAbstract.hpp.

32.12.2.3 virtual stdair::BomAbstract::~BomAbstract () `[inline, virtual]`

Destructor.

Definition at line 59 of file BomAbstract.hpp.

32.12.3 Member Function Documentation

32.12.3.1 virtual void stdair::BomAbstract::toStream (std::ostream & *ioOut*) const `[pure virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& The input/output stream.

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

32.12.3.2 virtual void stdair::BomAbstract::fromStream (std::istream & *ioIn*) [pure virtual]

Read a Business Object from an input stream.

Parameters:

istream& The input stream.

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

Referenced by operator>>().

32.12.3.3 virtual std::string stdair::BomAbstract::toString () const [pure virtual]

Get the serialised version of the Business Object.

Returns:

std::string The output string

Implemented in stdair::AirlineClassList, stdair::AirlineFeature, stdair::AirportPair, stdair::BomHolder< BOM >, stdair::BomRoot, stdair::BookingClass, stdair::Bucket, stdair::DatePeriod, stdair::FareFamily, stdair::FareFeatures, stdair::FlightDate, stdair::FlightPeriod, stdair::Inventory, stdair::LegCabin, stdair::LegDate, stdair::NestingNode, stdair::OnDDate, stdair::Policy, stdair::PosChannel, stdair::SegmentCabin, stdair::SegmentDate, stdair::SegmentPeriod, stdair::SegmentSnapshotTable, stdair::SimpleNestingStructure, stdair::TimePeriod, stdair::YieldFeatures, and stdair::YieldStore.

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

- stdair/bom/BomAbstract.hpp

32.13 stdair::BomArchive Class Reference

Utility class to archive/restore BOM objects with Boost serialisation.

```
#include <stdair/bom/BomArchive.hpp>
```

Static Public Member Functions

- static void archive (const BomRoot &)
- static std::string archive (const Inventory &)
- static void restore (const std::string &iArchive, Inventory &)
- static void archive (const FlightDate &)

32.13.1 Detailed Description

Utility class to archive/restore BOM objects with Boost serialisation.

Definition at line 28 of file BomArchive.hpp.

32.13.2 Member Function Documentation

32.13.2.1 void stdair::BomArchive::archive (const BomRoot & *iBomRoot*) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const BomRoot& Root of the BOM tree to be archived.

Definition at line 32 of file BomArchive.cpp.

32.13.2.2 std::string stdair::BomArchive::archive (const Inventory & *iInventory*) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const Inventory& Root of the BOM tree to be archived.

Definition at line 36 of file BomArchive.cpp.

32.13.2.3 void stdair::BomArchive::restore (const std::string & *iArchive*, Inventory & *ioInventory*) [static]

Recursively restore (from the underlying STL string) the objects of the BOM tree.

Parameters:

const std::string& String holding the serialised objects.

Inventory& Root of the BOM tree to be restored.

Definition at line 44 of file BomArchive.cpp.

32.13.2.4 void stdair::BomArchive::archive (const FlightDate & *iFlightDate*) [static]

Recursively archive (dump in the underlying STL string) the objects of the BOM tree.

Parameters:

const FlightDate& Root of the BOM tree to be archived.

Definition at line 52 of file BomArchive.cpp.

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

- stdair/bom/BomArchive.hpp
- stdair/bom/BomArchive.cpp

32.14 stdair::BomDisplay Class Reference

Utility class to display StdAir objects with a pretty format.

```
#include <stdair/bom/BomDisplay.hpp>
```

Static Public Member Functions

- static void list (std::ostream &, const BomRoot &, const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0)
- static void list (std::ostream &, const Inventory &, const unsigned short iInventoryIndex=0, const FlightNumber_T &iFlightNumber=0)
- static void listAirportPairDateRange (std::ostream &, const BomRoot &)
- static void csvDisplay (std::ostream &, const BomRoot &)
- static void csvDisplay (std::ostream &, const Inventory &)
- static void csvDisplay (std::ostream &, const OnDDate &)
- static void csvDisplay (std::ostream &, const FlightDate &)
- static void csvLegDateDisplay (std::ostream &, const FlightDate &)
- static void csvSegmentDateDisplay (std::ostream &, const FlightDate &)
- static void csvLegCabinDisplay (std::ostream &, const FlightDate &)
- static void csvSegmentCabinDisplay (std::ostream &, const FlightDate &)
- static void csvFareFamilyDisplay (std::ostream &, const FlightDate &)
- static void csvBucketDisplay (std::ostream &, const FlightDate &)
- static void csvBookingClassDisplay (std::ostream &, const BookingClass &, const std::string &iLeadingString)
- static void csvBookingClassDisplay (std::ostream &, const FlightDate &)
- static void csvDisplay (std::ostream &, const TravelSolutionList_T &)
- static void csvDisplay (std::ostream &, const DatePeriodList_T &)
- static void csvSimFQTAirRACDisplay (std::ostream &, const BomRoot &)
- static void csvAirportPairDisplay (std::ostream &, const AirportPair &)
- static void csvDateDisplay (std::ostream &, const DatePeriod &)
- static void csvPosChannelDisplay (std::ostream &, const PosChannel &)
- static void csvTimeDisplay (std::ostream &, const TimePeriod &)
- template<typename FEATURE_TYPE >
  static void csvFeatureListDisplay (std::ostream &oStream, const TimePeriod &)
- template<typename FEATURE_TYPE >
  static void csvFeaturesDisplay (std::ostream &oStream, const FEATURE_TYPE &)
- static void csvAirlineClassDisplay (std::ostream &, const AirlineClassList &)

32.14.1 Detailed Description

Utility class to display StdAir objects with a pretty format.

Definition at line 38 of file BomDisplay.hpp.

32.14.2 Member Function Documentation

32.14.2.1 static void stdair::BomDisplay::list (std::ostream &, const BomRoot &, const AirlineCode_T & *iAirlineCode* = "all", const FlightNumber_T & *iFlightNumber* = 0) [static]

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters:

std::ostream& Output stream in which the flight-date keys should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

32.14.2.2 static void stdair::BomDisplay::list (std::ostream &, const Inventory &, const unsigned short *iInventoryIndex* = 0, const FlightNumber_T & *iFlightNumber* = 0) [static]

Display (dump in the underlying output log stream) the list of flight-dates contained within the given BOM tree.

Parameters:

std::ostream& Output stream in which the flight-date keys should be logged/dumped.

const Inventory& Root of the BOM tree to be displayed.

const unsigned short Index, within the list, of the inventory. It is 0 when that inventory is displayed alone.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

32.14.2.3 static void stdair::BomDisplay::listAirportPairDateRange (std::ostream &, const BomRoot &) [static]

Display the list of aiports pairs and date ranges (contained within the BOM tree)

Parameters:

std::ostream& Output stream in which the airport pairs and date ranges are logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

32.14.2.4 static void stdair::BomDisplay::csvDisplay (std::ostream &, const BomRoot &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

32.14.2.5 static void stdair::BomDisplay::csvDisplay (std::ostream &, const Inventory &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given Inventory.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const Inventory& Root of the BOM tree to be displayed.

32.14.2.6 static void stdair::BomDisplay::csvDisplay (std::ostream &, const OnDDate &) [static]

Display the O&D date object information.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const OnDDate& the BOM to be displayed.

32.14.2.7 static void stdair::BomDisplay::csvDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given FlightDate.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.8 static void stdair::BomDisplay::csvLegDateDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the leg-date level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.9 static void stdair::BomDisplay::csvSegmentDateDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-date level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.10 static void stdair::BomDisplay::csvLegCabinDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the leg-cabin level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.11 static void stdair::BomDisplay::csvSegmentCabinDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-cabin level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.12 static void stdair::BomDisplay::csvFareFamilyDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the fare families level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.13 static void stdair::BomDisplay::csvBucketDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the bucket holder level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.14 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream &, const BookingClass &, const std::string & *iLeadingString*) [static]

Display (dump in the underlying output log stream) the segment-class, without going recursively deeper in the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BookingClass& Root of the BOM tree to be displayed.

const std::string& Leading string to be displayed.

32.14.2.15 static void stdair::BomDisplay::csvBookingClassDisplay (std::ostream &, const FlightDate &) [static]

Recursively display (dump in the underlying output log stream) the segment-class level objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be displayed.

32.14.2.16 static void stdair::BomDisplay::csvDisplay (std::ostream &, const TravelSolutionList_T &) [static]

Display (dump in the underlying output log stream) the full list of travel solution structures.

Parameters:

std::ostream& Output stream in which the list of travel solutions is logged/dumped.

TravelSolutionList_T& List of travel solutions to display.

32.14.2.17 static void stdair::BomDisplay::csvDisplay (std::ostream &, const DatePeriodList_T &) [static]

Display (dump in the underlying output log stream) the full list of date period fare rule sub bom tree.

Parameters:

std::ostream& Output stream in which the list of travel solutions is logged/dumped.

DatePeriodList_T& List of date period to display.

32.14.2.18 static void stdair::BomDisplay::csvSimFQTAirRACDisplay (std::ostream &, const BomRoot &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const BomRoot& Root of the BOM tree to be displayed.

32.14.2.19 static void stdair::BomDisplay::csvAirportPairDisplay (std::ostream &, const AirportPair &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given airport pair.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const AirportPair& Root of the BOM tree to be displayed.

32.14.2.20 static void stdair::BomDisplay::csvDateDisplay (std::ostream &, const DatePeriod &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given date range.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const DatePeriod& Root of the BOM tree to be displayed.

32.14.2.21 static void stdair::BomDisplay::csvPosChannelDisplay (std::ostream &, const PosChannel &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given point of sale channel.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const PosChannel& Root of the BOM tree to be displayed.

32.14.2.22 static void stdair::BomDisplay::csvTimeDisplay (std::ostream &, const TimePeriod &) [static]

Recursively display (dump in the underlying output log stream) the objects of the BOM tree from the level of the given time range.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const TimePeriod& Root of the BOM tree to be displayed.

32.14.2.23 template<typename FEATURE_TYPE > static void stdair::BomDisplay::csvFeatureListDisplay (std::ostream & *oStream*, const TimePeriod &) [inline, static]

Recursively display (dump in the underlying output log stream) the list of fare/yield features objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const TimePeriod& Root of the BOM tree to be displayed.

32.14.2.24 template<typename FEATURE_TYPE > static void stdair::BomDisplay::csvFeaturesDisplay (std::ostream & *oStream*, const FEATURE_TYPE &) `[inline, static]`

Recursively display (dump in the underlying output log stream) the fare/yield features objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FEATURE_TYPE& Root of the BOM tree to be displayed.

32.14.2.25 static void stdair::BomDisplay::csvAirlineClassDisplay (std::ostream &, const AirlineClassList &) `[static]`

Recursively display (dump in the underlying output log stream) the airline class objects of the BOM tree.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const AirlineClassList& Root of the BOM tree to be displayed.

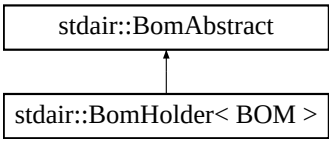
The documentation for this class was generated from the following file:

- stdair/bom/BomDisplay.hpp

32.15 stdair::BomHolder< BOM > Class Template Reference

Class representing the holder of BOM object containers (list and map).

`#include <stdair/bom/BomHolder.hpp>`Inheritance diagram for stdair::BomHolder< BOM >::



Public Types

- typedef stdair::BomHolderKey Key_T
- typedef std::list< BOM ∗ > BomList_T
- typedef std::map< const MapKey_T, BOM ∗ > BomMap_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Public Attributes

- Key_T _key
- BomList_T _bomList
- BomMap_T _bomMap

Protected Member Functions

- BomHolder ()
- BomHolder (const BomHolder &)
- BomHolder (const Key_T &iKey)
- ~BomHolder ()

Friends

- class FacBom
- class FacBomManager

32.15.1 Detailed Description

template<typename BOM> class stdair::BomHolder< BOM >

Class representing the holder of BOM object containers (list and map).

Definition at line 24 of file BomHolder.hpp.

32.15.2 Member Typedef Documentation

32.15.2.1 template<typename BOM> typedef stdair::BomHolderKey stdair::BomHolder< BOM >::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 34 of file BomHolder.hpp.

32.15.2.2 template<typename BOM> typedef std::list<BOM*> stdair::BomHolder< BOM >::BomList_T

(STL) list of children.

Definition at line 39 of file BomHolder.hpp.

32.15.2.3 template<typename BOM> typedef std::map<const MapKey_T, BOM*> stdair::BomHolder< BOM >::BomMap_T

(STL) map of children.

Definition at line 44 of file BomHolder.hpp.

32.15.3 Constructor & Destructor Documentation

32.15.3.1 template<typename BOM> stdair::BomHolder< BOM >::BomHolder () [protected]

Constructor.

32.15.3.2 template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const BomHolder< BOM > &) [protected]

Copy constructor.

32.15.3.3 template<typename BOM> stdair::BomHolder< BOM >::BomHolder (const Key_T & *iKey*) [inline, protected]

Main constructor.

Definition at line 94 of file BomHolder.hpp.

32.15.3.4 template<typename BOM> stdair::BomHolder< BOM >::~BomHolder () [inline, protected]

Destructor.

Definition at line 99 of file BomHolder.hpp.

32.15.4 Member Function Documentation

32.15.4.1 template<typename BOM> void stdair::BomHolder< BOM >::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 54 of file BomHolder.hpp.

References stdair::BomHolder< BOM >::toString().

32.15.4.2 template<typename BOM> void stdair::BomHolder< BOM >::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 63 of file BomHolder.hpp.

32.15.4.3 template<typename BOM> std::string stdair::BomHolder< BOM >::toString () const [inline, virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 69 of file BomHolder.hpp.

Referenced by stdair::BomHolder< BOM >::toStream().

32.15.4.4 template<typename BOM> const std::string stdair::BomHolder< BOM >::describeKey () const [inline]

Get a string describing the key.

Definition at line 76 of file BomHolder.hpp.

32.15.5 Friends And Related Function Documentation

32.15.5.1 template<typename BOM> friend class FacBom [friend]

Friend classes.

Definition at line 26 of file BomHolder.hpp.

32.15.5.2 template<typename BOM> friend class FacBomManager [friend]

Definition at line 27 of file BomHolder.hpp.

32.15.6 Member Data Documentation

32.15.6.1 template<typename BOM> Key_T stdair::BomHolder< BOM >::_key

Key.

Definition at line 99 of file BomHolder.hpp.

32.15.6.2 template<typename BOM> BomList_T stdair::BomHolder< BOM >::_bomList

(STL) list of children.

Definition at line 111 of file BomHolder.hpp.

Referenced by stdair::FacBomManager::cloneHolder(), stdair::BomManager::getList(), stdair::BomManager::hasList(), stdair::FacBomManager::resetYieldBasedNestingStructure(), and stdair::serialiseHelper().

32.15.6.3 template<typename BOM> BomMap_T stdair::BomHolder< BOM >::_bomMap

(STL) map of children.

Definition at line 116 of file BomHolder.hpp.

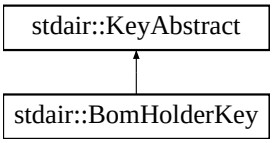
Referenced by stdair::FacBomManager::cloneHolder(), stdair::BomManager::getMap(), stdair::BomManager::getObjectPtr(), stdair::BomManager::hasMap(), and stdair::serialiseHelper().

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

- stdair/bom/BomHolder.hpp

32.16 stdair::BomHolderKey Struct Reference

`#include <stdair/bom/BomHolderKey.hpp>`Inheritance diagram for stdair::BomHolderKey::



Public Member Functions

- BomHolderKey ()
- ~BomHolderKey ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- const std::string describe () const

32.16.1 Detailed Description

Key of the BOM structure holder.

Definition at line 12 of file BomHolderKey.hpp.

32.16.2 Constructor & Destructor Documentation

32.16.2.1 stdair::BomHolderKey::BomHolderKey ()

Constructor.

Definition at line 13 of file BomHolderKey.cpp.

32.16.2.2 stdair::BomHolderKey::~BomHolderKey ()

Destructor.

Definition at line 17 of file BomHolderKey.cpp.

32.16.3 Member Function Documentation

32.16.3.1 void stdair::BomHolderKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 21 of file BomHolderKey.cpp.

References toString().

32.16.3.2 void stdair::BomHolderKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 26 of file BomHolderKey.cpp.

32.16.3.3 const std::string stdair::BomHolderKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file BomHolderKey.cpp.

Referenced by toStream().

32.16.3.4 const std::string stdair::BomHolderKey::describe () const

Display of the key.

The documentation for this struct was generated from the following files:

- stdair/bom/BomHolderKey.hpp
- stdair/bom/BomHolderKey.cpp

32.17 stdair::BomID< BOM > Struct Template Reference

Class wrapper of bom ID (e.g. pointer to object).

```
#include <stdair/bom/BomID.hpp>
```

Public Member Functions

- BOM & getObject () const
- BomID (BOM &iBOM)
- BomID (const BomID &)
- ~BomID ()

32.17.1 Detailed Description

template<typename BOM> struct stdair::BomID< BOM >

Class wrapper of bom ID (e.g. pointer to object).

Definition at line 17 of file BomID.hpp.

32.17.2 Constructor & Destructor Documentation

32.17.2.1 template<typename BOM > stdair::BomID< BOM >::BomID (BOM & *iBOM*) [inline]

Main constructor.

Definition at line 58 of file BomID.hpp.

32.17.2.2 template<typename BOM > stdair::BomID< BOM >::BomID (const BomID< BOM > & *iBomID*) [inline]

Copy constructor.

Definition at line 61 of file BomID.hpp.

32.17.2.3 template<typename BOM > stdair::BomID< BOM >::~BomID () [inline]

Destructor.

Definition at line 65 of file BomID.hpp.

32.17.3 Member Function Documentation

32.17.3.1 template<typename BOM > BOM & stdair::BomID< BOM >::getObject () const [inline]

Retrieve the object.

Definition at line 68 of file BomID.hpp.

Referenced by stdair::CancellationStruct::describe(), and stdair::CancellationStruct::display().

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

- stdair/bom/BomID.hpp

32.18 stdair::BomINIImport Class Reference

Utility class to import StdAir objects in a INI format.

`#include <stdair/bom/BomINIImport.hpp>`

Static Public Member Functions

- static void importINIConfig (ConfigHolderStruct &, const ConfigINIFile &)

32.18.1 Detailed Description

Utility class to import StdAir objects in a INI format.

Definition at line 21 of file BomINIImport.hpp.

32.18.2 Member Function Documentation

32.18.2.1 void stdair::BomINIImport::importINIConfig (ConfigHolderStruct & *iConfigHolder*, const ConfigINIFile & *iConfigINIFile*) `[static]`

Extract a boost property tree from an INI config file.

Parameters:

ConfigHolderStruct& Holder of the configuration tree.

const ConfigINIFile& INI config file.

Definition at line 29 of file BomINIImport.cpp.

References stdair::ConfigHolderStruct::add(), stdair::BasFileMgr::doesExistAndIsReadable(), stdair::RootFilePath::name(), and STDAIR_LOG_DEBUG.

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

- stdair/bom/BomINIImport.hpp
- stdair/bom/BomINIImport.cpp

32.19 stdair::BomJSONExport Class Reference

Utility class to export StdAir objects in a JSON format.

`#include <stdair/bom/BomJSONExport.hpp>`

Static Public Member Functions

- static void jsonExportFlightDateList (std::ostream &, const BomRoot &, const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0)
- static void jsonExportFlightDateObjects (std::ostream &, const FlightDate &)
- static void jsonExportBookingRequestObject (std::ostream &, const EventStruct &)
- static void jsonExportBreakPointObject (std::ostream &, const EventStruct &)

32.19.1 Detailed Description

Utility class to export StdAir objects in a JSON format.

Definition at line 42 of file BomJSONExport.hpp.

32.19.2 Member Function Documentation

32.19.2.1 void stdair::BomJSONExport::jsonExportFlightDateList (std::ostream & *oStream*, const BomRoot & *iBomRoot*, const AirlineCode_T & *iAirlineCode* = `"all"`, const FlightNumber_T & *iFlightNumber* = `0`) `[static]`

Export (dump in the underlying output log stream and in JSON format) a list of flight date objects.

Parameters:

std::ostream& Output stream in which the flight date objects should be logged/dumped.

const BomRoot& Root of the BOM tree containing flight-dates to be exported.

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 35 of file BomJSONExport.cpp.

References stdair::Inventory::getAirlineCode(), stdair::FlightDate::getDepartureDate(), and stdair::FlightDate::getFlightNumber().

32.19.2.2 void stdair::BomJSONExport::jsonExportFlightDateObjects (std::ostream & *oStream*, const FlightDate & *iFlightDate*) `[static]`

Recursively export (dump in the underlying output log stream and in JSON format) the objects of the BOM tree using the given FlightDate as root.

Parameters:

std::ostream& Output stream in which the BOM tree should be logged/dumped.

const FlightDate& Root of the BOM tree to be exported.

Definition at line 163 of file BomJSONExport.cpp.

References stdair::FlightDate::getAirlineCode(), stdair::FlightDate::getDepartureDate(), and stdair::FlightDate::getFlightNumber().

32.19.2.3 void stdair::BomJSONExport::jsonExportBookingRequestObject (std::ostream & *oStream*, const EventStruct & *iEventStruct*) `[static]`

Export (dump in the underlying output log stream and in JSON format) the booking request object contained in the event structure.

Parameters:

std::ostream& Output stream in which the events should be logged/dumped.

const EventStruct& Booking request to be stored in JSON-ified format.

Definition at line 660 of file BomJSONExport.cpp.

References stdair::EventType::BKG_REQ, stdair::BookingRequestStruct::getBookingChannel(), stdair::EventStruct::getBookingRequest(), stdair::BookingRequestStruct::getDestination(), stdair::EventStruct::getEventType(), stdair::EventType::getLabel(), stdair::BookingRequestStruct::getOrigin(), stdair::BookingRequestStruct::getPartySize(), stdair::BookingRequestStruct::getPOS(), stdair::BookingRequestStruct::getPreferedDepartureDate(), stdair::BookingRequestStruct::getPreferredCabin(), stdair::BookingRequestStruct::getPreferredDepartureTime(), stdair::BookingRequestStruct::getRequestDateTime(), stdair::BookingRequestStruct::getStayDuration(), and stdair::BookingRequestStruct::getWTP().

Referenced by stdair::STDAIR_Service::jsonExportEventObject().

32.19.2.4 void stdair::BomJSONExport::jsonExportBreakPointObject (std::ostream & *oStream*, const EventStruct & *iEventStruct*) `[static]`

Export (dump in the underlying output log stream and in JSON format) the break point object contained in the event structure.

Parameters:

std::ostream& Output stream in which the events should be logged/dumped.

const EventStruct& Break point to be stored in JSON-ified format.

Definition at line 749 of file BomJSONExport.cpp.

References stdair::EventType::BRK_PT, stdair::EventStruct::getBreakPoint(), stdair::BreakPointStruct::getBreakPointTime(), stdair::EventStruct::getEventType(), and stdair::EventType::getLabel().

Referenced by stdair::STDAIR_Service::jsonExportEventObject().

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

- stdair/bom/BomJSONExport.hpp
- stdair/bom/BomJSONExport.cpp

32.20 stdair::BomJSONImport Class Reference

Utility class to import StdAir objects in a JSON format.

```
#include <stdair/bom/BomJSONImport.hpp>
```

Static Public Member Functions

- static bool jsonImportCommand (const JSONString &, JSonCommand::EN_JSonCommand &)
- static bool jsonImportInventoryKey (const JSONString &, AirlineCode_T &)
- static bool jsonImportFlightDate (const JSONString &, Date_T &)
- static bool jsonImportFlightNumber (const JSONString &, FlightNumber_T &)
- static bool jsonImportBreakPoints (const JSONString &, BreakPointList_T &)
- static bool jsonImportEventType (const JSONString &, EventType::EN_EventType &)
- static bool jsonImportConfig (const JSONString &, ConfigHolderStruct &)

32.20.1 Detailed Description

Utility class to import StdAir objects in a JSON format.

Definition at line 26 of file BomJSONImport.hpp.

32.20.2 Member Function Documentation

32.20.2.1 bool stdair::BomJSONImport::jsonImportCommand (const JSONString & *iBomJSONStr*, JSonCommand::EN_JSonCommand & *ioEnumJSonCommand*) [static]

Extract the JSON command from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

JSonCommand::EN_JSonCommand& JSOM command extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 32 of file BomJSONImport.cpp.

References stdair::JSonCommand::getCommand(), and stdair::JSONString::getString().

32.20.2.2 bool stdair::BomJSONImport::jsonImportInventoryKey (const JSONString & *iBomJSONStr*, AirlineCode_T & *ioAirlineCode*) [static]

Extract the airline code from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

AirlineCode_T& Airline code extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 98 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

32.20.2.3 bool stdair::BomJSONImport::jsonImportFlightDate (const JSONString & *iBomJSONStr*, Date_T & *ioDepartureDate*) [static]

Extract the FlightDate from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

Date_T& Departure date extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 133 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

32.20.2.4 bool stdair::BomJSONImport::jsonImportFlightNumber (const JSONString & *iBomJSONStr*, FlightNumber_T & *ioFlightNumber*) [static]

Extract the FlightNumber from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

FlightNumber_T& Flight number extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 167 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

32.20.2.5 bool stdair::BomJSONImport::jsonImportBreakPoints (const JSONString & *iBomJSONStr*, BreakPointList_T & *oBreakPointList*) [static]

Extract the break points from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

BreakPointList_T& List of breaking points extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 203 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

32.20.2.6 bool stdair::BomJSONImport::jsonImportEventType (const JSONString & *iBomJSONStr*, EventType::EN_EventType & *ioEventType*) [static]

Extract the event type from a given JSON-formatted string.

Parameters:

const JSONString& JSON-formatted string.

EventType::EN_EventType& Event type extracted from the given string.

Returns:

bool State whether the extracting has been successful.

Definition at line 253 of file BomJSONImport.cpp.

References stdair::JSONString::getString().

32.20.2.7 bool stdair::BomJSONImport::jsonImportConfig (const JSONString & *iBomJSONStr*, ConfigHolderStruct & *iConfigHolderStruct*) `[static]`

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder

Parameters:

const JSONString& JSON-formatted string.

ConfigHolderStruct& Configuration holder.

Returns:

bool State whether the extracting has been successful.

Definition at line 296 of file BomJSONImport.cpp.

References stdair::ConfigHolderStruct::add(), and stdair::JSONString::getString().

Referenced by stdair::STDAIR_Service::jsonImportConfiguration().

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

- stdair/bom/BomJSONImport.hpp
- stdair/bom/BomJSONImport.cpp

32.21 stdair::BomKeyManager Class Reference

Utility class to extract key structures from strings.

`#include <stdair/bom/BomKeyManager.hpp>`

Static Public Member Functions

- static ParsedKey extractKeys (const std::string &iFullKeyStr)
- static InventoryKey extractInventoryKey (const std::string &iFullKeyStr)
- static FlightDateKey extractFlightDateKey (const std::string &iFullKeyStr)
- static SegmentDateKey extractSegmentDateKey (const std::string &iFullKeyStr)
- static LegDateKey extractLegDateKey (const std::string &iFullKeyStr)

32.21.1 Detailed Description

Utility class to extract key structures from strings.

Definition at line 29 of file BomKeyManager.hpp.

32.21.2 Member Function Documentation

32.21.2.1 ParsedKey stdair::BomKeyManager::extractKeys (const std::string & *iFullKeyStr*) [static]

Build a ParsedKey structure from a full key string which includes an inventory key, flight-date key elements, segment-date key elements.

Definition at line 31 of file BomKeyManager.cpp.

References stdair::ParsedKey::_airlineCode, stdair::ParsedKey::_boardingPoint, stdair::ParsedKey::_boardingTime, stdair::ParsedKey::_departureDate, stdair::ParsedKey::_flightNumber, stdair::ParsedKey::_fullKey, stdair::ParsedKey::_offPoint, and stdair::DEFAULT_KEY_TOKEN_DELIMITER.

Referenced by stdair::TravelSolutionStruct::describe(), stdair::TravelSolutionStruct::describeSegmentPath(), stdair::TravelSolutionStruct::display(), extractFlightDateKey(), extractInventoryKey(), extractLegDateKey(), extractSegmentDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.21.2.2 InventoryKey stdair::BomKeyManager::extractInventoryKey (const std::string & *iFullKeyStr*) [static]

Build a InventoryKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

InventoryKey The just built InventoryKey structure.

Definition at line 79 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getInventoryKey().

Referenced by stdair::BomRetriever::retrieveInventoryFromLongKey(), and stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey().

32.21.2.3 FlightDateKey stdair::BomKeyManager::extractFlightDateKey (const std::string & *iFullKeyStr*) [static]

Build a FlightDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

FlightDateKey The just built FlightDateKey structure.

Definition at line 87 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getFlightDateKey().

Referenced by stdair::OnDDateKey::getDate(), and stdair::BomRetriever::retrieveFlightDateFromLongKey().

32.21.2.4 SegmentDateKey stdair::BomKeyManager::extractSegmentDateKey (const std::string & *iFullKeyStr*) `[static]`

Build a SegmentDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

SegmentDateKey The just built SegmentDateKey structure.

Definition at line 95 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getSegmentKey().

Referenced by stdair::OnDDateKey::getDestination(), stdair::OnDDateKey::getOrigin(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.21.2.5 LegDateKey stdair::BomKeyManager::extractLegDateKey (const std::string & *iFullKeyStr*) `[static]`

Build a LegDateKey structure from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const std::string& The full key string.

Returns:

LegDateKey The just built LegDateKey structure.

Definition at line 103 of file BomKeyManager.cpp.

References extractKeys(), and stdair::ParsedKey::getLegKey().

Referenced by stdair::BomRetriever::retrieveOperatingLegDateFromLongKey().

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

- stdair/bom/BomKeyManager.hpp
- stdair/bom/BomKeyManager.cpp

32.22 stdair::BomManager Class Reference

Utility class for StdAir-based objects.

```
#include <stdair/bom/BomManager.hpp>
```

Public Member Functions

- template<>
  bool hasList (const SegmentDate &ioSegmentDate)
- template<>
  const BomHolder< SegmentDate >::BomList_T & getList (const SegmentDate &ioSegmentDate)
- template<>
  bool hasMap (const SegmentDate &ioSegmentDate)
- template<>
  bool hasList (const Inventory &ioInventory)
- template<>
  bool hasMap (const Inventory &ioInventory)
- template<>
  AirlineFeature * getObjectPtr (const Inventory &iInventory, const MapKey_T &iKey)
- template<>
  AirlineFeature & getObject (const Inventory &iInventory, const MapKey_T &iKey)

Static Public Member Functions

- template<typename OBJECT2 , typename OBJECT1 >
  static const BomHolder< OBJECT2 >::BomList_T & getList (const OBJECT1 &)
- template<typename OBJECT2 , typename OBJECT1 >
  static const BomHolder< OBJECT2 >::BomMap_T & getMap (const OBJECT1 &)
- template<typename OBJECT2 , typename OBJECT1 >
  static bool hasList (const OBJECT1 &)
- template<typename OBJECT2 , typename OBJECT1 >
  static bool hasMap (const OBJECT1 &)
- template<typename PARENT , typename CHILD >
  static PARENT * getParentPtr (const CHILD &)
- template<typename PARENT , typename CHILD >
  static PARENT & getParent (const CHILD &)
- template<typename OBJECT2 , typename OBJECT1 >
  static OBJECT2 * getObjectPtr (const OBJECT1 &, const MapKey_T &)
- template<typename OBJECT2 , typename OBJECT1 >
  static OBJECT2 & getObject (const OBJECT1 &, const MapKey_T &)

Friends

- class FacBomManager

32.22.1 Detailed Description

Utility class for StdAir-based objects. Most of those methods work for objects specified and instantianted outside StdAir, as long as those objects inherit from StdAir objects.

Definition at line 34 of file BomManager.hpp.

32.22.2 Member Function Documentation

32.22.2.1 template<typename OBJECT2 , typename OBJECT1 > const BomHolder< OBJECT2 >::BomList_T & stdair::BomManager::getList (const OBJECT1 & *iObject1*) [inline, static]

Get the container (STL list) of OBJECT2 objects within the OBJECT1 object.

Definition at line 140 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomList.

32.22.2.2 template<typename OBJECT2 , typename OBJECT1 > const BomHolder< OBJECT2 >::BomMap_T & stdair::BomManager::getMap (const OBJECT1 & *iObject1*) [inline, static]

Get the container (STL map) of OBJECT2 objects within the OBJECT1 object.

Definition at line 159 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomMap.

32.22.2.3 template<typename OBJECT2 , typename OBJECT1 > bool stdair::BomManager::hasList (const OBJECT1 & *iObject1*) [inline, static]

Check if the list of object2 has been initialised.

Definition at line 181 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomList.

32.22.2.4 template<typename OBJECT2 , typename OBJECT1 > bool stdair::BomManager::hasMap (const OBJECT1 & *iObject1*) [inline, static]

Check if the map of object2 has been initialised.

Definition at line 201 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomMap.

32.22.2.5 template<typename PARENT , typename CHILD > PARENT * stdair::BomManager::getParentPtr (const CHILD & *iChild*) [inline, static]

Get the PARENT of the given CHILD.

If the types do not match, NULL is returned.

Definition at line 220 of file BomManager.hpp.

32.22.2.6 template<typename PARENT , typename CHILD > PARENT & stdair::BomManager::getParent (const CHILD & *iChild*) [inline, static]

Get the PARENT of the given CHILD.

Definition at line 230 of file BomManager.hpp.

32.22.2.7 template<typename OBJECT2 , typename OBJECT1 > OBJECT2 * stdair::BomManager::getObjectPtr (const OBJECT1 & *iObject1*, const MapKey_T & *iKey*) [inline, static]

Get the OBJECT2 pointer corresponding to the given string key.

If such a OBJECT2 does not exist, return NULL.

Definition at line 241 of file BomManager.hpp.

References stdair::BomHolder< BOM >::_bomMap.

32.22.2.8 template<typename OBJECT2 , typename OBJECT1 > OBJECT2 & stdair::BomManager::getObject (const OBJECT1 & *iObject1*, const MapKey_T & *iKey*) [inline, static]

Get the OBJECT2 corresponding to the given string key.

Definition at line 283 of file BomManager.hpp.

References STDAIR_LOG_ERROR.

32.22.2.9 template<> bool stdair::BomManager::hasList (const SegmentDate & *ioSegmentDate*) [inline]

32.22.2.10 template<> const BomHolder<SegmentDate>::BomList_T& stdair::BomManager::getList (const SegmentDate & *ioSegmentDate*) [inline]

32.22.2.11 template<> bool stdair::BomManager::hasMap (const SegmentDate & *ioSegmentDate*) [inline]

32.22.2.12 template<> bool stdair::BomManager::hasList (const Inventory & *ioInventory*) [inline]

32.22.2.13 template<> bool stdair::BomManager::hasMap (const Inventory & *ioInventory*) [inline]

32.22.2.14 template<> AirlineFeature* stdair::BomManager::getObjectPtr (const Inventory & *iInventory*, const MapKey_T & *iKey*) [inline]

32.22.2.15 template<> AirlineFeature& stdair::BomManager::getObject (const Inventory & *iInventory*, const MapKey_T & *iKey*) [inline]

32.22.3 Friends And Related Function Documentation

32.22.3.1 friend class FacBomManager [friend]

Definition at line 35 of file BomManager.hpp.

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

- stdair/bom/BomManager.hpp

32.23 stdair::BomRetriever Class Reference

Utility class to retrieve StdAir objects.

```
#include <stdair/bom/BomRetriever.hpp>
```

Static Public Member Functions

- static Inventory * retrieveInventoryFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static Inventory * retrieveInventoryFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static Inventory * retrieveInventoryFromKey (const BomRoot &, const InventoryKey &)
- static Inventory * retrieveInventoryFromKey (const BomRoot &, const AirlineCode_T &)
- static AirlineFeature * retrieveAirlineFeatureFromKey (const BomRoot &, const AirlineCode_T &)
- static FlightDate * retrieveFlightDateFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static FlightDate * retrieveFlightDateFromKeySet (const BomRoot &, const AirlineCode_T &, const FlightNumber_T &, const Date_T &iFlightDateDate)
- static FlightDate * retrieveFlightDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static FlightDate * retrieveFlightDateFromKey (const Inventory &, const FlightDateKey &)
- static FlightDate * retrieveFlightDateFromKey (const Inventory &, const FlightNumber_T &, const Date_T &iFlightDateDate)
- static LegDate * retrieveOperatingLegDateFromLongKey (const FlightDate &, const std::string &iFullKeyStr)
- static SegmentDate * retrievePartnerSegmentDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const BomRoot &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const Inventory &, const std::string &iFullKeyStr)
- static SegmentDate * retrieveSegmentDateFromLongKey (const FlightDate &, const std::string &iFullKeyStr)

- static SegmentDate * retrieveSegmentDateFromKey (const FlightDate &, const SegmentDateKey &)
- static SegmentDate * retrieveSegmentDateFromKey (const FlightDate &, const AirportCode_T &iOrigin, const AirportCode_T &iDestination)
- static BookingClass * retrieveBookingClassFromLongKey (const Inventory &, const std::string &iFullKeyStr, const ClassCode_T &)
- static AirportPair * retrieveAirportPairFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &)
- static void retrieveDatePeriodListFromKey (const AirportPair &, const stdair::Date_T &, stdair::DatePeriodList_T &)
- static void retrieveDatePeriodListFromKeySet (const BomRoot &, const stdair::AirportCode_T &, const stdair::AirportCode_T &, const stdair::Date_T &, stdair::DatePeriodList_T &)
- static stdair::LegCabin & retrieveDummyLegCabin (stdair::BomRoot &, const bool isForFareFamilies=false)
- static stdair::SegmentCabin & retrieveDummySegmentCabin (stdair::BomRoot &, const bool isForFareFamilies=false)
- static std::string retrieveFullKeyFromSegmentDate (const SegmentDate &)

32.23.1 Detailed Description

Utility class to retrieve StdAir objects.

Definition at line 36 of file BomRetriever.hpp.

32.23.2 Member Function Documentation

32.23.2.1 Inventory * stdair::BomRetriever::retrieveInventoryFromLongKey (const BomRoot & *iBomRoot*, const std::string & *iFullKeyStr*) [static]

Retrieve an Inventory object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

Inventory* The just retrieved Inventory object.

Definition at line 31 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), and stdair::BomRoot::getInventory().

Referenced by retrieveFlightDateFromLongKey(), and retrievePartnerSegmentDateFromLongKey().

32.23.2.2 Inventory * stdair::BomRetriever::retrieveInventoryFromLongKey (const Inventory & *iInventory*, const std::string & *iFullKeyStr*) [static]

Retrieve an Inventory object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

Inventory∗ The just retrieved Inventory object.

Definition at line 46 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), and stdair::InventoryKey::getAirlineCode().

32.23.2.3 Inventory ∗ stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot & *iBomRoot*, const InventoryKey & *iKey*) [static]

Retrieve an Inventory object from an InventoryKey structure.

Parameters:

const BomRoot& The root of the BOM tree.

const InventoryKey& The key.

Returns:

Inventory∗ The just retrieved Inventory object.

Definition at line 63 of file BomRetriever.cpp.

References stdair::BomRoot::getInventory().

Referenced by retrieveAirlineFeatureFromKey(), retrieveDummyLegCabin(), retrieveDummySegmentCabin(), and retrieveFlightDateFromKeySet().

32.23.2.4 Inventory ∗ stdair::BomRetriever::retrieveInventoryFromKey (const BomRoot & *iBomRoot*, const AirlineCode_T & *iAirlineCode*) [static]

Retrieve an Inventory object from an InventoryKey structure.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

Returns:

Inventory∗ The just retrieved Inventory object.

Definition at line 75 of file BomRetriever.cpp.

References stdair::BomRoot::getInventory().

32.23.2.5 AirlineFeature * stdair::BomRetriever::retrieveAirlineFeatureFromKey (const BomRoot & *iBomRoot*, const AirlineCode_T & *iAirlineCode*) [static]

Retrieve an Airline Feature object from an airline code.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

Returns:

AirlineFeature* The just retrieved Airline Feature object.

Definition at line 88 of file BomRetriever.cpp.

References retrieveInventoryFromKey().

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.23.2.6 FlightDate * stdair::BomRetriever::retrieveFlightDateFromLongKey (const BomRoot & *iBomRoot*, const std::string & *iFullKeyStr*) [static]

Retrieve a FlightDate object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 109 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), stdair::Inventory::getFlightDate(), and retrieveInventoryFromLongKey().

Referenced by retrieveSegmentDateFromLongKey().

32.23.2.7 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKeySet (const BomRoot & *iBomRoot*, const AirlineCode_T & *iAirlineCode*, const FlightNumber_T & *iFlightNumber*, const Date_T & *iFlightDateDate*) [static]

Retrieve a FlightDate object from a set of keys.

Parameters:

const BomRoot& The root of the BOM tree.

const AirlineCode_T& The key.

const FlightNumber_T& Part of the key.

const Date_T& Part of the key.

Returns:

FlightDate∗ The just retrieved FlightDate object.

Definition at line 132 of file BomRetriever.cpp.

References retrieveFlightDateFromKey(), and retrieveInventoryFromKey().

Referenced by stdair::STDAIR_Service::check(), stdair::STDAIR_Service::csvDisplay(), and stdair::STDAIR_Service::jsonExportFlightDateObjects().

32.23.2.8 FlightDate ∗ stdair::BomRetriever::retrieveFlightDateFromLongKey (const Inventory & *iInventory*, const std::string & *iFullKeyStr*) [static]

Retrieve a FlightDate object from a (full) key string.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

FlightDate∗ The just retrieved FlightDate object.

Definition at line 155 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), and stdair::Inventory::getFlightDate().

32.23.2.9 FlightDate ∗ stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory & *iInventory*, const FlightDateKey & *iKey*) [static]

Retrieve a FlightDate object from an FlightDateKey structure.

Parameters:

const Inventory& The root of the BOM tree.

const FlightDateKey& The key.

Returns:

FlightDate∗ The just retrieved FlightDate object.

Definition at line 170 of file BomRetriever.cpp.

References stdair::Inventory::getFlightDate().

Referenced by retrieveDummyLegCabin(), retrieveDummySegmentCabin(), retrieveFlightDateFromKeySet(), and retrieveSegmentDateFromLongKey().

32.23.2.10 FlightDate * stdair::BomRetriever::retrieveFlightDateFromKey (const Inventory & *iInventory*, const FlightNumber_T & *iFlightNumber*, const Date_T & *iFlightDateDate*) [static]

Retrieve a FlightDate object from an FlightDateKey structure.

Parameters:

const Inventory& The root of the BOM tree.

const FlightNumber_T& Part of the key.

const Date_T& Part of the key.

Returns:

FlightDate* The just retrieved FlightDate object.

Definition at line 182 of file BomRetriever.cpp.

References stdair::Inventory::getFlightDate().

32.23.2.11 LegDate * stdair::BomRetriever::retrieveOperatingLegDateFromLongKey (const FlightDate & *iFlightDate*, const std::string & *iFullKeyStr*) [static]

Retrieve a LegDate object from an FlightDate structure.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const FlightDate& The root of the BOM tree.

const std::string& The full key string.

Returns:

LegDate* The just retrieved LegDate object.

Definition at line 266 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractLegDateKey(), and stdair::FlightDate::getLegDate().

32.23.2.12 SegmentDate * stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey (const Inventory & *iInventory*, const std::string & *iFullKeyStr*) [static]

Retrieve a partner SegmentDate object from an Inventory structure.

The full key string gathers airline code, flight number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate∗ The just retrieved SegmentDate object.

Definition at line 281 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractInventoryKey(), stdair::InventoryKey::getAirlineCode(), retrieveInventoryFromLongKey(), and retrieveSegmentDateFromLongKey().

32.23.2.13 SegmentDate ∗ stdair::BomRetriever::retrieveSegmentDateFromLongKey (const BomRoot & *iBomRoot*, const std::string & *iFullKeyStr*) `[static]`

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const BomRoot& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate∗ The just retrieved SegmentDate object.

Definition at line 196 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), stdair::FlightDate::getSegmentDate(), and retrieveFlightDateFromLongKey().

Referenced by retrieveBookingClassFromLongKey(), and retrievePartnerSegmentDateFromLongKey().

32.23.2.14 SegmentDate ∗ stdair::BomRetriever::retrieveSegmentDateFromLongKey (const Inventory & *iInventory*, const std::string & *iFullKeyStr*) `[static]`

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate∗ The just retrieved SegmentDate object.

Definition at line 219 of file BomRetriever.cpp.

References stdair::ParsedKey::_airlineCode, stdair::BomKeyManager::extractKeys(), stdair::Inventory::getAirlineCode(), stdair::ParsedKey::getFlightDateKey(), stdair::ParsedKey::getSegmentKey(), retrieveFlightDateFromKey(), retrieveSegmentDateFromKey(), STDAIR_LOG_DEBUG, stdair::SegmentDateKey::toString(), and stdair::FlightDateKey::toString().

32.23.2.15 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromLongKey (const FlightDate & *iFlightDate*, const std::string & *iFullKeyStr*) [static]

Retrieve a SegmentDate object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Parameters:

const FlightDate& The root of the BOM tree.

const std::string& The full key string.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 251 of file BomRetriever.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::FlightDate::getSegmentDate().

32.23.2.16 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate & *iFlightDate*, const SegmentDateKey & *iKey*) [static]

Retrieve a SegmentDate object from an SegmentDateKey structure.

Parameters:

const FlightDate& The root of the BOM tree.

const SegmentDateKey& The key.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 307 of file BomRetriever.cpp.

References stdair::FlightDate::getSegmentDate().

Referenced by retrieveSegmentDateFromLongKey().

32.23.2.17 SegmentDate * stdair::BomRetriever::retrieveSegmentDateFromKey (const FlightDate & *iFlightDate*, const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*) [static]

Retrieve a SegmentDate object from an SegmentDateKey structure.

Parameters:

const FlightDate& The root of the BOM tree.

const AirportCode_T& Origin, part of the key.

const AirportCode_T& Destination, part of the key.

Returns:

SegmentDate* The just retrieved SegmentDate object.

Definition at line 319 of file BomRetriever.cpp.

References stdair::FlightDate::getSegmentDate().

32.23.2.18 BookingClass * stdair::BomRetriever::retrieveBookingClassFromLongKey (const Inventory & *iInventory*, const std::string & *iFullKeyStr*, const ClassCode_T & *iClassCode*) [static]

Retrieve a BookingClass object from a (full) key string.

The full key string gathers airline code, segment number, origin and destination, cabin and booking class. It corresponds to the output generated by the toString() methods of the XxxKey structures.

Besides being attached to segment-cabin objects (and fare family objects, when they exist), the booking-class objects must also be attached directly to the segment-date.

Hence, if an assertion fails within that method call, chances are that the booking-class objects have not been attached to the segment-date objects. Check, for instance, the CmdBomManager::buildSampleBom() to see how that should be properly done.

Parameters:

const Inventory& The root of the BOM tree.

const std::string& Part of the full key string.

const ClassCode_T& Part of the full key string.

Returns:

BookingClass* The just retrieved BookingClass object.

Definition at line 333 of file BomRetriever.cpp.

References retrieveSegmentDateFromLongKey().

32.23.2.19 AirportPair * stdair::BomRetriever::retrieveAirportPairFromKeySet (const BomRoot & *iBomRoot*, const stdair::AirportCode_T & *iOrigin*, const stdair::AirportCode_T & *iDestination*) [static]

Retrieve an AirportPair object from an AirportPair structure.

Parameters:

const BomRoot& The root of the BOM tree.

const AirportCode_T& Origin, part of the key.

const AirportCode_T& Destination, part of the key.

Returns:

AirportPair* The just retrieved AirportPair object.

Definition at line 355 of file BomRetriever.cpp.

References stdair::AirportPairKey::toString().

Referenced by retrieveDatePeriodListFromKeySet().

32.23.2.20 void stdair::BomRetriever::retrieveDatePeriodListFromKey (const AirportPair & *iAirportPair*, const stdair::Date_T & *iDepartureDate*, stdair::DatePeriodList_T & *ioDatePeriodList*) `[static]`

Retrieve a list of date-period corresponding to a flight date.

Parameters:

const AirportPair& The root of the BOM tree.

const Date_T& Departure Date of the flight

stdair::DatePeriodList_T& List of DatePeriod to display.

Definition at line 373 of file BomRetriever.cpp.

References stdair::DatePeriod::isDepartureDateValid().

Referenced by retrieveDatePeriodListFromKeySet().

32.23.2.21 void stdair::BomRetriever::retrieveDatePeriodListFromKeySet (const BomRoot & *iBomRoot*, const stdair::AirportCode_T & *iOrigin*, const stdair::AirportCode_T & *iDestination*, const stdair::Date_T & *iDepartureDate*, stdair::DatePeriodList_T & *ioDatePeriodList*) `[static]`

Retrieve a list of date-period from a set of keys.

Parameters:

const BomRoot& The root of the BOM tree.

const AirportCode_T& Part of the AirportPair key: the origin airport

const AirportCode_T& Part of the AirportPair key: the destination airport.

const Date_T& Departure date of the flight

stdair::DatePeriodList_T& List of DatePeriod to display.

Definition at line 404 of file BomRetriever.cpp.

References retrieveAirportPairFromKeySet(), and retrieveDatePeriodListFromKey().

Referenced by stdair::STDAIR_Service::check(), and stdair::STDAIR_Service::csvDisplay().

32.23.2.22 LegCabin & stdair::BomRetriever::retrieveDummyLegCabin (stdair::BomRoot & *iBomRoot*, const bool *isForFareFamilies* = `false`) `[static]`

Retrieve one sample leg-cabin of the dummy inventory of "XX".

Parameters:

stdair::BomRoot& The BOM tree.

const bool Boolean to choose the sample leg-cabin. True: the dummy leg-cabin with fare families. False: the dummy leg-cabin without fare families. By default the value is false.

Definition at line 427 of file BomRetriever.cpp.

References stdair::DEFAULT_AIRLINE_CODE, stdair::DEFAULT_CABIN_CODE, stdair::DEFAULT_-DEPARTURE_DATE, stdair::DEFAULT_FLIGHT_NUMBER, stdair::DEFAULT_FLIGHT_NUMBER_-FF, stdair::DEFAULT_ORIGIN, stdair::LegDate::getLegCabin(), stdair::FlightDate::getLegDate(), re-trieveFlightDateFromKey(), and retrieveInventoryFromKey().

32.23.2.23 SegmentCabin & stdair::BomRetriever::retrieveDummySegmentCabin (stdair::BomRoot & *iBomRoot*, const bool *isForFareFamilies* = `false`) `[static]`

Retrieve one sample segment-cabin of the dummy inventory of "XX".

Parameters:

stdair::BomRoot& The BOM tree.

const bool Boolean to choose the sample segment-cabin. True: the dummy segment-cabin with fare families. False: the dummy segment-cabin without fare families. By default the value is false.

Definition at line 502 of file BomRetriever.cpp.

References stdair::DEFAULT_AIRLINE_CODE, stdair::DEFAULT_CABIN_CODE, stdair::DEFAULT_DEPARTURE_DATE, stdair::DEFAULT_DESTINATION, stdair::DEFAULT_-FLIGHT_NUMBER, stdair::DEFAULT_FLIGHT_NUMBER_FF, stdair::DEFAULT_ORIGIN, stdair::FlightDate::getSegmentDate(), retrieveFlightDateFromKey(), retrieveInventoryFromKey(), and stdair::SegmentCabinKey::toString().

32.23.2.24 std::string stdair::BomRetriever::retrieveFullKeyFromSegmentDate (const SegmentDate & *iSegmentdate*) `[static]`

Retrieve the whole key of the segment date, that is to say a string composed of the inventory key, the flight date key and the segment date key.

Parameters:

const SegmentDate& Segment date to retrieve the whole key for.

Returns:

std::string The just retrieved whole key.

Definition at line 578 of file BomRetriever.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER, stdair::SegmentDate::describeKey(), and stdair::Inventory::describeKey().

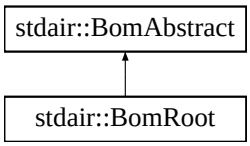
The documentation for this class was generated from the following files:

- stdair/bom/BomRetriever.hpp
- stdair/bom/BomRetriever.cpp

32.24 stdair::BomRoot Class Reference

Class representing the actual attributes for the Bom root.

`#include <stdair/bom/BomRoot.hpp>`Inheritance diagram for stdair::BomRoot::



Public Types

- typedef BomRootKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const HolderMap_T & getHolderMap () const
- const FRAT5Curve_T & getFRAT5Curve (const std::string &iKey) const
- const FFDisutilityCurve_T & getFFDisutilityCurve (const std::string &iKey) const
- Inventory * getInventory (const std::string &iInventoryKeyStr) const
- Inventory * getInventory (const InventoryKey &) const
- void addFRAT5Curve (const std::string &iKey, const FRAT5Curve_T &iCurve)
- void addFFDisutilityCurve (const std::string &iKey, const FFDisutilityCurve_T &iCurve)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- BomRoot ()
- BomRoot (const BomRoot &)
- BomRoot (const Key_T &iKey)
- ~BomRoot ()

Protected Attributes

- Key_T _key
- HolderMap_T _holderMap
- FRAT5CurveHolderStruct _frat5CurveHolder
- FFDisutilityCurveHolderStruct _ffDisutilityCurveHolder

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.24.1 Detailed Description

Class representing the actual attributes for the Bom root.

Definition at line 32 of file BomRoot.hpp.

32.24.2 Member Typedef Documentation

32.24.2.1 typedef BomRootKey stdair::BomRoot::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 42 of file BomRoot.hpp.

32.24.3 Constructor & Destructor Documentation

32.24.3.1 stdair::BomRoot::BomRoot () [protected]

Default constructor.

Definition at line 17 of file BomRoot.cpp.

32.24.3.2 stdair::BomRoot::BomRoot (const BomRoot & *iBomRoot*) [protected]

Copy constructor.

Definition at line 22 of file BomRoot.cpp.

32.24.3.3 stdair::BomRoot::BomRoot (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 28 of file BomRoot.cpp.

32.24.3.4 stdair::BomRoot::~BomRoot () [protected]

Destructor.

Definition at line 32 of file BomRoot.cpp.

32.24.4 Member Function Documentation

32.24.4.1 const Key_T& stdair::BomRoot::getKey () const [inline]

Get the inventory key (airline code).

Definition at line 48 of file BomRoot.hpp.

References _key.

32.24.4.2 const HolderMap_T& stdair::BomRoot::getHolderMap () const [inline]

Get the map of children.

Definition at line 53 of file BomRoot.hpp.

References _holderMap.

32.24.4.3 const FRAT5Curve_T& stdair::BomRoot::getFRAT5Curve (const std::string & *iKey*) const [inline]

Get the FRAT5 curve corresponding to the given key.

Definition at line 58 of file BomRoot.hpp.

References _frat5CurveHolder, and stdair::FRAT5CurveHolderStruct::getFRAT5Curve().

32.24.4.4 const FFDisutilityCurve_T& stdair::BomRoot::getFFDisutilityCurve (const std::string & *iKey*) const [inline]

Get the FFDisutility curve corresponding to the given key.

Definition at line 63 of file BomRoot.hpp.

References _ffDisutilityCurveHolder, and stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve().

32.24.4.5 Inventory * stdair::BomRoot::getInventory (const std::string & *iInventoryKeyStr*) const

Get a pointer on the Inventory object corresponding to the given key.

Note:

The Inventory object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The flight-date key.

Returns:

Inventory* Found Inventory object. NULL if not found.

Definition at line 43 of file BomRoot.cpp.

Referenced by getInventory(), stdair::BomRetriever::retrieveInventoryFromKey(), and stdair::BomRetriever::retrieveInventoryFromLongKey().

32.24.4.6 Inventory * stdair::BomRoot::getInventory (const InventoryKey & *iInventoryKey*) const

Get a pointer on the Inventory object corresponding to the given key.

Note:

The Inventory object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const InventoryKey& The flight-date key

Returns:

Inventory* Found Inventory object. NULL if not found.

Definition at line 50 of file BomRoot.cpp.

References getInventory(), and stdair::InventoryKey::toString().

32.24.4.7 void stdair::BomRoot::addFRAT5Curve (const std::string & *iKey*, const FRAT5Curve_T & *iCurve*) [inline]

Add a new FRAT5 curve to the holder.

Definition at line 93 of file BomRoot.hpp.

References _frat5CurveHolder, and stdair::FRAT5CurveHolderStruct::addCurve().

32.24.4.8 void stdair::BomRoot::addFFDisutilityCurve (const std::string & *iKey*, const FFDisutilityCurve_T & *iCurve*) [inline]

Add a new FF disutility curve to the holder.

Definition at line 98 of file BomRoot.hpp.

References _ffDisutilityCurveHolder, and stdair::FFDisutilityCurveHolderStruct::addCurve().

32.24.4.9 void stdair::BomRoot::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 111 of file BomRoot.hpp.

References toString().

32.24.4.10 void stdair::BomRoot::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 120 of file BomRoot.hpp.

32.24.4.11 std::string stdair::BomRoot::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file BomRoot.cpp.

References _key, and stdair::BomRootKey::toString().

Referenced by toStream().

32.24.4.12 const std::string stdair::BomRoot::describeKey () const [inline]

Get a string describing the key.

Definition at line 131 of file BomRoot.hpp.

References _key, and stdair::BomRootKey::toString().

32.24.4.13 template<class Archive > void stdair::BomRoot::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

That method is used both for serialisation a BOM tree (into a backup file/stream), as well as re-instantiating a BOM tree from a back-up file/stream.

Note:

The implementation of that method is to be found in the CmdBomSerialiser command.

Definition at line 133 of file CmdBomSerialiser.cpp.

References _key.

32.24.5 Friends And Related Function Documentation

32.24.5.1 friend class FacBom [friend]

Definition at line 33 of file BomRoot.hpp.

32.24.5.2 friend class FacCloneBom [friend]

Definition at line 34 of file BomRoot.hpp.

32.24.5.3 friend class FacBomManager [friend]

Definition at line 35 of file BomRoot.hpp.

32.24.5.4 friend class boost::serialization::access [friend]

Definition at line 36 of file BomRoot.hpp.

32.24.6 Member Data Documentation

32.24.6.1 Key_T stdair::BomRoot::_key [protected]

Primary key.

Definition at line 191 of file BomRoot.hpp.

Referenced by describeKey(), getKey(), serialize(), and toString().

32.24.6.2 HolderMap_T stdair::BomRoot::_holderMap [protected]

Map holding the children (Inventory objects).

Definition at line 196 of file BomRoot.hpp.

Referenced by getHolderMap().

32.24.6.3 FRAT5CurveHolderStruct stdair::BomRoot::_frat5CurveHolder [protected]

Holder of FRAT5 curves.

Definition at line 201 of file BomRoot.hpp.

Referenced by addFRAT5Curve(), and getFRAT5Curve().

32.24.6.4 FFDisutilityCurveHolderStruct stdair::BomRoot::_ffDisutilityCurveHolder [protected]

Holder of fare family disutility curves.

Definition at line 206 of file BomRoot.hpp.

Referenced by addFFDisutilityCurve(), and getFFDisutilityCurve().

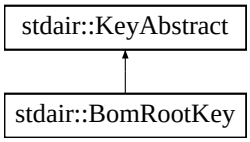
The documentation for this class was generated from the following files:

- stdair/bom/BomRoot.hpp
- stdair/bom/BomRoot.cpp
- stdair/command/CmdBomSerialiser.cpp

32.25 stdair::BomRootKey Struct Reference

Key of the BOM structure root.

`#include <stdair/bom/BomRootKey.hpp>`Inheritance diagram for stdair::BomRootKey::



Public Member Functions

- BomRootKey ()
- BomRootKey (const std::string &iIdentification)
- BomRootKey (const BomRootKey &)
- ~BomRootKey ()

- const std::string & getID () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.25.1 Detailed Description

Key of the BOM structure root.

Definition at line 25 of file BomRootKey.hpp.

32.25.2 Constructor & Destructor Documentation

32.25.2.1 stdair::BomRootKey::BomRootKey ()

Default constructor.

Definition at line 18 of file BomRootKey.cpp.

32.25.2.2 stdair::BomRootKey::BomRootKey (const std::string & *iIdentification*)

Constructor.

Definition at line 28 of file BomRootKey.cpp.

32.25.2.3 stdair::BomRootKey::BomRootKey (const BomRootKey & *iBomRootKey*)

Copy constructor.

Definition at line 23 of file BomRootKey.cpp.

32.25.2.4 stdair::BomRootKey::~BomRootKey ()

Destructor.

Definition at line 33 of file BomRootKey.cpp.

32.25.3 Member Function Documentation

32.25.3.1 const std::string& stdair::BomRootKey::getID () const `[inline]`

Get the identification.

Definition at line 56 of file BomRootKey.hpp.

32.25.3.2 void stdair::BomRootKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file BomRootKey.cpp.

References toString().

32.25.3.3 void stdair::BomRootKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file BomRootKey.cpp.

32.25.3.4 const std::string stdair::BomRootKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file BomRootKey.cpp.

Referenced by stdair::BomRoot::describeKey(), toStream(), and stdair::BomRoot::toString().

32.25.3.5 template<class Archive > void stdair::BomRootKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 68 of file BomRootKey.cpp.

32.25.4 Friends And Related Function Documentation

32.25.4.1 friend class boost::serialization::access [friend]

Definition at line 26 of file BomRootKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/BomRootKey.hpp
- stdair/bom/BomRootKey.cpp

32.26 stdair_test::BookingClass Struct Reference

```
#include <test/stdair/StdairTestLib.hpp>
```

Public Member Functions

- BookingClass (const std::string &iClassCode)
- std::string toString () const

Public Attributes

- std::string _classCode

32.26.1 Detailed Description

BookingClass

Definition at line 16 of file StdairTestLib.hpp.

32.26.2 Constructor & Destructor Documentation

32.26.2.1 stdair_test::BookingClass::BookingClass (const std::string & *iClassCode*) [inline]

Constructor.

Definition at line 19 of file StdairTestLib.hpp.

32.26.3 Member Function Documentation

32.26.3.1 std::string stdair_test::BookingClass::toString () const [inline]

Display .

Definition at line 24 of file StdairTestLib.hpp.

References _classCode.

32.26.4 Member Data Documentation

32.26.4.1 std::string stdair_test::BookingClass::_classCode

Definition at line 17 of file StdairTestLib.hpp.

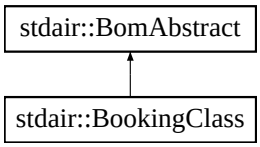
Referenced by stdair_test::Cabin::toString(), and toString().

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

- test/stdair/StdairTestLib.hpp

32.27 stdair::BookingClass Class Reference

`#include <stdair/bom/BookingClass.hpp>`Inheritance diagram for stdair::BookingClass::



Public Types

- typedef BookingClassKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const ClassCode_T & getClassCode () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const SubclassCode_T & getSubclassCode () const
- const AuthorizationLevel_T & getAuthorizationLevel () const
- const ProtectionLevel_T & getProtection () const
- const ProtectionLevel_T & getCumulatedProtection () const
- const BookingLimit_T & getCumulatedBookingLimit () const
- const NbOfSeats_T & getNegotiatedSpace () const
- const OverbookingRate_T & getNoShowPercentage () const
- const OverbookingRate_T & getCancellationPercentage () const
- const NbOfBookings_T & getNbOfBookings () const
- const NbOfBookings_T & getNbOfGroupBookings () const
- const NbOfBookings_T & getNbOfPendingGroupBookings () const
- const NbOfBookings_T & getNbOfStaffBookings () const
- const NbOfBookings_T & getNbOfWLBookings () const
- const NbOfCancellations_T & getNbOfCancellations () const
- const NbOfBookings_T & getETB () const
- const Availability_T & getNetClassAvailability () const
- const Availability_T & getSegmentAvailability () const
- const Availability_T & getNetRevenueAvailability () const
- const Yield_T & getYield () const
- const Yield_T & getAdjustedYield () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const MeanValue_T & getPriceDemMean () const
- const StdDevValue_T & getPriceDemStdDev () const
- const MeanValue_T & getCumuPriceDemMean () const
- const StdDevValue_T & getCumuPriceDemStdDev () const
- const MeanValue_T & getProductDemMean () const
- const StdDevValue_T & getProductDemStdDev () const
- const GeneratedDemandVector_T & getGeneratedDemandVector () const

- void setCumulatedProtection (const ProtectionLevel_T &iPL)
- void setProtection (const ProtectionLevel_T &iPL)
- void setCumulatedBookingLimit (const BookingLimit_T &iBL)
- void setAuthorizationLevel (const AuthorizationLevel_T &iAU)
- void setSegmentAvailability (const Availability_T &iAvl)
- void setYield (const Yield_T &iYield)
- void setAdjustedYield (const Yield_T &iYield)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setPriceDemMean (const MeanValue_T &iMean)
- void setPriceDemStdDev (const StdDevValue_T &iStdDev)
- void setCumuPriceDemMean (const MeanValue_T &iMean)
- void setCumuPriceDemStdDev (const StdDevValue_T &iStdDev)
- void setProductDemMean (const MeanValue_T &iMean)
- void setProductDemStdDev (const StdDevValue_T &iStdDev)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- void sell (const NbOfBookings_T &)
- void cancel (const NbOfBookings_T &)
- void generateDemandSamples (const NbOfSamples_T &)
- void generateDemandSamples (const NbOfSamples_T &, const RandomSeed_T &)

Protected Member Functions

- BookingClass (const Key_T &)
- virtual ~BookingClass ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SubclassCode_T _subclassCode
- ProtectionLevel_T _cumulatedProtection
- ProtectionLevel_T _protection
- BookingLimit_T _cumulatedBookingLimit
- AuthorizationLevel_T _au
- NbOfSeats_T _nego
- OverbookingRate_T _noShowPercentage
- OverbookingRate_T _cancellationPercentage
- NbOfBookings_T _nbOfBookings
- NbOfBookings_T _groupNbOfBookings
- NbOfBookings_T _groupPendingNbOfBookings
- NbOfBookings_T _staffNbOfBookings
- NbOfBookings_T _wlNbOfBookings
- NbOfCancellations_T _nbOfCancellations
- NbOfBookings_T _etb
- Availability_T _netClassAvailability

- Availability_T _segmentAvailability
- Availability_T _netRevenueAvailability
- Yield_T _yield
- Yield_T _adjustedYield
- MeanValue_T _mean
- StdDevValue_T _stdDev
- MeanValue_T _priceDemMean
- StdDevValue_T _priceDemStdDev
- MeanValue_T _cumuPriceDemMean
- StdDevValue_T _cumuPriceDemStdDev
- MeanValue_T _productDemMean
- StdDevValue_T _productDemStdDev
- GeneratedDemandVector_T _generatedDemandVector

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.27.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 24 of file BookingClass.hpp.

32.27.2 Member Typedef Documentation

32.27.2.1 typedef BookingClassKey stdair::BookingClass::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 32 of file BookingClass.hpp.

32.27.3 Constructor & Destructor Documentation

32.27.3.1 stdair::BookingClass::BookingClass (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 49 of file BookingClass.cpp.

32.27.3.2 stdair::BookingClass::~BookingClass () `[protected, virtual]`

Destructor.

Definition at line 61 of file BookingClass.cpp.

32.27.4 Member Function Documentation

32.27.4.1 const Key_T& stdair::BookingClass::getKey () const [inline]

Get the booking class key.

Definition at line 37 of file BookingClass.hpp.

References _key.

32.27.4.2 const ClassCode_T& stdair::BookingClass::getClassCode () const [inline]

Get the booking code (part of the primary key).

Definition at line 42 of file BookingClass.hpp.

References _key, and stdair::BookingClassKey::getClassCode().

Referenced by stdair::CancellationStruct::describe(), and stdair::CancellationStruct::display().

32.27.4.3 BomAbstract* const stdair::BookingClass::getParent () const [inline]

Get the parent object.

Definition at line 47 of file BookingClass.hpp.

References _parent.

32.27.4.4 const HolderMap_T& stdair::BookingClass::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 52 of file BookingClass.hpp.

References _holderMap.

32.27.4.5 const SubclassCode_T& stdair::BookingClass::getSubclassCode () const [inline]

Get teh sub-class code.

Definition at line 57 of file BookingClass.hpp.

References _subclassCode.

32.27.4.6 const AuthorizationLevel_T& stdair::BookingClass::getAuthorizationLevel () const [inline]

Get the authorisation level (AU, i.e., cumulated protection).

Definition at line 62 of file BookingClass.hpp.

References _au.

32.27.4.7 const ProtectionLevel_T& stdair::BookingClass::getProtection () const [inline]

Get the protection.

Definition at line 67 of file BookingClass.hpp.

References _protection.

32.27.4.8 const ProtectionLevel_T& stdair::BookingClass::getCumulatedProtection () const [inline]

Get the cumulated protection.

Definition at line 72 of file BookingClass.hpp.

References _cumulatedProtection.

32.27.4.9 const BookingLimit_T& stdair::BookingClass::getCumulatedBookingLimit () const [inline]

Get the cumulated booking limit.

Definition at line 77 of file BookingClass.hpp.

References _cumulatedBookingLimit.

32.27.4.10 const NbOfSeats_T& stdair::BookingClass::getNegotiatedSpace () const [inline]

Get the negotiated space.

Definition at line 82 of file BookingClass.hpp.

References _nego.

32.27.4.11 const OverbookingRate_T& stdair::BookingClass::getNoShowPercentage () const [inline]

Get the no-show rate.

Definition at line 87 of file BookingClass.hpp.

References _noShowPercentage.

32.27.4.12 const OverbookingRate_T& stdair::BookingClass::getCancellationPercentage () const [inline]

Get the cancellation rate.

Definition at line 92 of file BookingClass.hpp.

References _cancellationPercentage.

32.27.4.13 const NbOfBookings_T& stdair::BookingClass::getNbOfBookings () const [inline]

Get the number of bookings.

Definition at line 97 of file BookingClass.hpp.

References _nbOfBookings.

32.27.4.14 const NbOfBookings_T& stdair::BookingClass::getNbOfGroupBookings () const [inline]

Get the number of group bookings.

Definition at line 102 of file BookingClass.hpp.

References _groupNbOfBookings.

32.27.4.15 const NbOfBookings_T& stdair::BookingClass::getNbOfPendingGroupBookings () const [inline]

Get the number of pending group bookings.

Definition at line 107 of file BookingClass.hpp.

References _groupPendingNbOfBookings.

32.27.4.16 const NbOfBookings_T& stdair::BookingClass::getNbOfStaffBookings () const [inline]

Get the number of staff bookings.

Definition at line 112 of file BookingClass.hpp.

References _staffNbOfBookings.

32.27.4.17 const NbOfBookings_T& stdair::BookingClass::getNbOfWLBookings () const [inline]

Get the number of wait-list bookings.

Definition at line 117 of file BookingClass.hpp.

References _wlNbOfBookings.

32.27.4.18 const NbOfCancellations_T& stdair::BookingClass::getNbOfCancellations () const [inline]

Get the number of cancellations.

Definition at line 122 of file BookingClass.hpp.

References _nbOfCancellations.

32.27.4.19 const NbOfBookings_T& stdair::BookingClass::getETB () const [inline]

Get the expected number of passengers to board (ETB).

Definition at line 127 of file BookingClass.hpp.

References _etb.

32.27.4.20 const Availability_T& stdair::BookingClass::getNetClassAvailability () const [inline]

Get the net segment class availability.

Definition at line 132 of file BookingClass.hpp.

References _netClassAvailability.

32.27.4.21 const Availability_T& stdair::BookingClass::getSegmentAvailability () const [inline]

Get the segment class availability.

Definition at line 137 of file BookingClass.hpp.

References _segmentAvailability.

32.27.4.22 const Availability_T& stdair::BookingClass::getNetRevenueAvailability () const [inline]

Net revenue availability.

Definition at line 142 of file BookingClass.hpp.

References _netRevenueAvailability.

32.27.4.23 const Yield_T& stdair::BookingClass::getYield () const [inline]

Yield.

Definition at line 147 of file BookingClass.hpp.

References _yield.

32.27.4.24 const Yield_T& stdair::BookingClass::getAdjustedYield () const [inline]

Definition at line 148 of file BookingClass.hpp.

References _adjustedYield.

32.27.4.25 const MeanValue_T& stdair::BookingClass::getMean () const [inline]

Demand distribution.

Definition at line 151 of file BookingClass.hpp.

References _mean.

32.27.4.26 const StdDevValue_T& stdair::BookingClass::getStdDev () const [inline]

Definition at line 152 of file BookingClass.hpp.

References _stdDev.

32.27.4.27 const MeanValue_T& stdair::BookingClass::getPriceDemMean () const [inline]

Definition at line 153 of file BookingClass.hpp.

References _priceDemMean.

32.27.4.28 const StdDevValue_T& stdair::BookingClass::getPriceDemStdDev () const [inline]

Definition at line 154 of file BookingClass.hpp.

References _priceDemStdDev.

32.27.4.29 const MeanValue_T& stdair::BookingClass::getCumuPriceDemMean () const [inline]

Definition at line 155 of file BookingClass.hpp.

References _cumuPriceDemMean.

32.27.4.30 const StdDevValue_T& stdair::BookingClass::getCumuPriceDemStdDev () const [inline]

Definition at line 158 of file BookingClass.hpp.

References _cumuPriceDemStdDev.

32.27.4.31 const MeanValue_T& stdair::BookingClass::getProductDemMean () const [inline]

Definition at line 161 of file BookingClass.hpp.

References _productDemMean.

32.27.4.32 const StdDevValue_T& stdair::BookingClass::getProductDemStdDev () const [inline]

Definition at line 162 of file BookingClass.hpp.

References _productDemStdDev.

32.27.4.33 const GeneratedDemandVector_T& stdair::BookingClass::getGeneratedDemandVector () const [inline]

Generated demand vector.

Definition at line 165 of file BookingClass.hpp.

References _generatedDemandVector.

Referenced by stdair::VirtualClassStruct::getGeneratedDemandVector().

32.27.4.34 void stdair::BookingClass::setCumulatedProtection (const ProtectionLevel_T & *iPL*) [inline]

Cumulated protection.

Definition at line 172 of file BookingClass.hpp.

References _cumulatedProtection.

32.27.4.35 void stdair::BookingClass::setProtection (const ProtectionLevel_T & *iPL*) [inline]

Protection.

Definition at line 177 of file BookingClass.hpp.

References _protection.

32.27.4.36 void stdair::BookingClass::setCumulatedBookingLimit (const BookingLimit_T & *iBL*) [inline]

Cumulated booking limit.

Definition at line 182 of file BookingClass.hpp.

References _cumulatedBookingLimit.

32.27.4.37 void stdair::BookingClass::setAuthorizationLevel (const AuthorizationLevel_T & *iAU*) [inline]

Authorization level.

Definition at line 187 of file BookingClass.hpp.

References _au.

32.27.4.38 void stdair::BookingClass::setSegmentAvailability (const Availability_T & *iAvl*) [inline]

Set availability.

Definition at line 192 of file BookingClass.hpp.

References _segmentAvailability.

32.27.4.39 void stdair::BookingClass::setYield (const Yield_T & *iYield*) [inline]

Yield.

Definition at line 197 of file BookingClass.hpp.

References _adjustedYield, and _yield.

32.27.4.40 void stdair::BookingClass::setAdjustedYield (const Yield_T & *iYield*) [inline]

Definition at line 201 of file BookingClass.hpp.

References _adjustedYield.

32.27.4.41 void stdair::BookingClass::setMean (const MeanValue_T & *iMean*) [inline]

Demand distribution.

Definition at line 204 of file BookingClass.hpp.

References _mean.

32.27.4.42 void stdair::BookingClass::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 205 of file BookingClass.hpp.

References _stdDev.

32.27.4.43 void stdair::BookingClass::setPriceDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 206 of file BookingClass.hpp.

References _priceDemMean.

32.27.4.44 void stdair::BookingClass::setPriceDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 207 of file BookingClass.hpp.

References _priceDemStdDev.

32.27.4.45 void stdair::BookingClass::setCumuPriceDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 210 of file BookingClass.hpp.

References _cumuPriceDemMean.

32.27.4.46 void stdair::BookingClass::setCumuPriceDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 212 of file BookingClass.hpp.

References _cumuPriceDemStdDev.

32.27.4.47 void stdair::BookingClass::setProductDemMean (const MeanValue_T & *iMean*) [inline]

Definition at line 215 of file BookingClass.hpp.

References _productDemMean.

32.27.4.48 void stdair::BookingClass::setProductDemStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 218 of file BookingClass.hpp.

References _productDemStdDev.

32.27.4.49 void stdair::BookingClass::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 226 of file BookingClass.hpp.

References toString().

32.27.4.50 void stdair::BookingClass::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 232 of file BookingClass.hpp.

32.27.4.51 std::string stdair::BookingClass::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 65 of file BookingClass.cpp.

References describeKey().

Referenced by toStream().

32.27.4.52 const std::string stdair::BookingClass::describeKey () const [inline]

Get a string describing the key.

Definition at line 239 of file BookingClass.hpp.

References _key, and stdair::BookingClassKey::toString().

Referenced by toString().

32.27.4.53 void stdair::BookingClass::sell (const NbOfBookings_T & *iNbOfBookings*)

Register a sale.

Definition at line 72 of file BookingClass.cpp.

References _nbOfBookings.

32.27.4.54 void stdair::BookingClass::cancel (const NbOfBookings_T & *iNbOfCancellations*)

Register a cancellation.

Definition at line 77 of file BookingClass.cpp.

References _nbOfBookings, and _nbOfCancellations.

32.27.4.55 void stdair::BookingClass::generateDemandSamples (const NbOfSamples_T & *K*)

Generate demand samples for Monte-Carlo method with the default random seed.

Definition at line 83 of file BookingClass.cpp.

References _generatedDemandVector, _mean, _stdDev, stdair::DEFAULT_RANDOM_SEED, and stdair::RandomGeneration::generateNormal().

32.27.4.56 void stdair::BookingClass::generateDemandSamples (const NbOfSamples_T & *K*, const RandomSeed_T & *iSeed*)

Generate demand samples for Monte-Carlo method with the given random seed.

Definition at line 95 of file BookingClass.cpp.

References _generatedDemandVector, _mean, _stdDev, and stdair::RandomGeneration::generateNormal().

32.27.5 Friends And Related Function Documentation

32.27.5.1 friend class FacBom [friend]

Definition at line 25 of file BookingClass.hpp.

32.27.5.2 friend class FacCloneBom [friend]

Definition at line 26 of file BookingClass.hpp.

32.27.5.3 friend class FacBomManager [friend]

Definition at line 27 of file BookingClass.hpp.

32.27.6 Member Data Documentation

32.27.6.1 Key_T stdair::BookingClass::_key [protected]

Primary key (booking class code).

Definition at line 276 of file BookingClass.hpp.

Referenced by describeKey(), getClassCode(), and getKey().

32.27.6.2 BomAbstract* stdair::BookingClass::_parent [protected]

Pointer on the parent class (SegmentCabin).

Definition at line 279 of file BookingClass.hpp.

Referenced by getParent().

32.27.6.3 HolderMap_T stdair::BookingClass::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 282 of file BookingClass.hpp.

Referenced by getHolderMap().

32.27.6.4 SubclassCode_T stdair::BookingClass::_subclassCode [protected]

Sub-class code.

Definition at line 285 of file BookingClass.hpp.

Referenced by getSubclassCode().

32.27.6.5 ProtectionLevel_T stdair::BookingClass::_cumulatedProtection [protected]

Cumulated protection.

Definition at line 288 of file BookingClass.hpp.

Referenced by getCumulatedProtection(), and setCumulatedProtection().

32.27.6.6 ProtectionLevel_T stdair::BookingClass::_protection [protected]

Protection.

Definition at line 291 of file BookingClass.hpp.

Referenced by getProtection(), and setProtection().

32.27.6.7 BookingLimit_T stdair::BookingClass::_cumulatedBookingLimit [protected]

Cumulated booking limit.

Definition at line 294 of file BookingClass.hpp.

Referenced by getCumulatedBookingLimit(), and setCumulatedBookingLimit().

32.27.6.8 AuthorizationLevel_T stdair::BookingClass::_au [protected]

Authorization level.

Definition at line 297 of file BookingClass.hpp.

Referenced by getAuthorizationLevel(), and setAuthorizationLevel().

32.27.6.9 NbOfSeats_T stdair::BookingClass::_nego [protected]

Negotiated space.

Definition at line 300 of file BookingClass.hpp.

Referenced by getNegotiatedSpace().

32.27.6.10 OverbookingRate_T stdair::BookingClass::_noShowPercentage [protected]

Overbooking rate.

Definition at line 303 of file BookingClass.hpp.

Referenced by getNoShowPercentage().

32.27.6.11 OverbookingRate_T stdair::BookingClass::_cancellationPercentage [protected]

Cancellation rate.

Definition at line 306 of file BookingClass.hpp.

Referenced by getCancellationPercentage().

32.27.6.12 NbOfBookings_T stdair::BookingClass::_nbOfBookings [protected]

Number of bookings.

Definition at line 309 of file BookingClass.hpp.

Referenced by cancel(), getNbOfBookings(), and sell().

32.27.6.13 NbOfBookings_T stdair::BookingClass::_groupNbOfBookings [protected]

Number of group bookings.

Definition at line 312 of file BookingClass.hpp.

Referenced by getNbOfGroupBookings().

32.27.6.14 NbOfBookings_T stdair::BookingClass::_groupPendingNbOfBookings [protected]

Number of pending group bookings.

Definition at line 315 of file BookingClass.hpp.

Referenced by getNbOfPendingGroupBookings().

32.27.6.15 NbOfBookings_T stdair::BookingClass::_staffNbOfBookings [protected]

Number of staff bookings.

Definition at line 318 of file BookingClass.hpp.

Referenced by getNbOfStaffBookings().

32.27.6.16 NbOfBookings_T stdair::BookingClass::_wlNbOfBookings [protected]

Number of wait-list bookings.

Definition at line 321 of file BookingClass.hpp.

Referenced by getNbOfWLBookings().

32.27.6.17 NbOfCancellations_T stdair::BookingClass::_nbOfCancellations [protected]

Number of cancellations.

Definition at line 324 of file BookingClass.hpp.

Referenced by cancel(), and getNbOfCancellations().

32.27.6.18 NbOfBookings_T stdair::BookingClass::_etb [protected]

Expected to board (ETB).

Definition at line 327 of file BookingClass.hpp.

Referenced by getETB().

32.27.6.19 Availability_T stdair::BookingClass::_netClassAvailability [protected]

Net segment class availability.

Definition at line 330 of file BookingClass.hpp.

Referenced by getNetClassAvailability().

32.27.6.20 Availability_T stdair::BookingClass::_segmentAvailability [protected]

Segment class availability.

Definition at line 333 of file BookingClass.hpp.

Referenced by getSegmentAvailability(), and setSegmentAvailability().

32.27.6.21 Availability_T stdair::BookingClass::_netRevenueAvailability [protected]

Net revenue availability.

Definition at line 336 of file BookingClass.hpp.

Referenced by getNetRevenueAvailability().

32.27.6.22 Yield_T stdair::BookingClass::_yield [protected]

Yield.

Definition at line 339 of file BookingClass.hpp.

Referenced by getYield(), and setYield().

32.27.6.23 Yield_T stdair::BookingClass::_adjustedYield [protected]

Definition at line 340 of file BookingClass.hpp.

Referenced by getAdjustedYield(), setAdjustedYield(), and setYield().

32.27.6.24 MeanValue_T stdair::BookingClass::_mean [protected]

Demand distribution forecast.

Definition at line 343 of file BookingClass.hpp.

Referenced by generateDemandSamples(), getMean(), and setMean().

32.27.6.25 StdDevValue_T stdair::BookingClass::_stdDev [protected]

Definition at line 344 of file BookingClass.hpp.

Referenced by generateDemandSamples(), getStdDev(), and setStdDev().

32.27.6.26 MeanValue_T stdair::BookingClass::_priceDemMean [protected]

Price-oriented demand distribution forecast.

Definition at line 347 of file BookingClass.hpp.

Referenced by getPriceDemMean(), and setPriceDemMean().

32.27.6.27 StdDevValue_T stdair::BookingClass::_priceDemStdDev [protected]

Definition at line 348 of file BookingClass.hpp.

Referenced by getPriceDemStdDev(), and setPriceDemStdDev().

32.27.6.28 MeanValue_T stdair::BookingClass::_cumuPriceDemMean [protected]

Cumulative price-oriented demand distribution forecast.

Definition at line 351 of file BookingClass.hpp.

Referenced by getCumuPriceDemMean(), and setCumuPriceDemMean().

32.27.6.29 StdDevValue_T stdair::BookingClass::_cumuPriceDemStdDev [protected]

Definition at line 352 of file BookingClass.hpp.

Referenced by getCumuPriceDemStdDev(), and setCumuPriceDemStdDev().

32.27.6.30 MeanValue_T stdair::BookingClass::_productDemMean [protected]

Product-oriented demand distribution forecast.

Definition at line 355 of file BookingClass.hpp.

Referenced by getProductDemMean(), and setProductDemMean().

32.27.6.31 StdDevValue_T stdair::BookingClass::_productDemStdDev [protected]

Definition at line 356 of file BookingClass.hpp.

Referenced by getProductDemStdDev(), and setProductDemStdDev().

32.27.6.32 GeneratedDemandVector_T stdair::BookingClass::_generatedDemandVector [protected]

Vector of number of demand samples drawn from the demand distribution.

Definition at line 359 of file BookingClass.hpp.

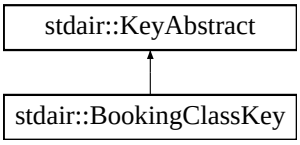
Referenced by generateDemandSamples(), and getGeneratedDemandVector().

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

- stdair/bom/BookingClass.hpp
- stdair/bom/BookingClass.cpp

32.28 stdair::BookingClassKey Struct Reference

`#include <stdair/bom/BookingClassKey.hpp>`Inheritance diagram for stdair::BookingClassKey::



Public Member Functions

- BookingClassKey (const ClassCode_T &iClassCode)
- BookingClassKey (const BookingClassKey &)
- ~BookingClassKey ()
- const ClassCode_T & getClassCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.28.1 Detailed Description

Key of a given leg-cabin, made of a cabin code.

Definition at line 16 of file BookingClassKey.hpp.

32.28.2 Constructor & Destructor Documentation

32.28.2.1 stdair::BookingClassKey::BookingClassKey (const ClassCode_T & *iClassCode*)

Constructor.

Definition at line 24 of file BookingClassKey.cpp.

32.28.2.2 stdair::BookingClassKey::BookingClassKey (const BookingClassKey & *iKey*)

Default copy constructor.

Definition at line 19 of file BookingClassKey.cpp.

32.28.2.3 stdair::BookingClassKey::~BookingClassKey ()

Destructor.

Definition at line 29 of file BookingClassKey.cpp.

32.28.3 Member Function Documentation

32.28.3.1 const ClassCode_T& stdair::BookingClassKey::getClassCode () const [inline]

Get the class code.

Definition at line 34 of file BookingClassKey.hpp.

Referenced by stdair::BookingClass::getClassCode().

32.28.3.2 void stdair::BookingClassKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 33 of file BookingClassKey.cpp.

References toString().

32.28.3.3 void stdair::BookingClassKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file BookingClassKey.cpp.

32.28.3.4 const std::string stdair::BookingClassKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file BookingClassKey.cpp.

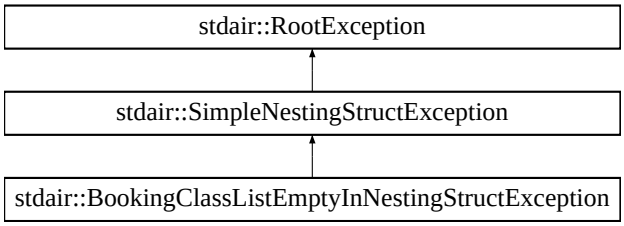
Referenced by stdair::BookingClass::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/BookingClassKey.hpp
- stdair/bom/BookingClassKey.cpp

32.29 stdair::BookingClassListEmptyInNestingStructException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::BookingClassListEmptyInNestingStructException::



Public Member Functions

- BookingClassListEmptyInNestingStructException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.29.1 Detailed Description

Empty booking class list in Simple Nesting Structure.

Definition at line 219 of file stdair_exceptions.hpp.

32.29.2 Constructor & Destructor Documentation

32.29.2.1 stdair::BookingClassListEmptyInNestingStructException::BookingClassListEmptyInNestingStructExceptio (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 223 of file stdair_exceptions.hpp.

32.29.3 Member Function Documentation

32.29.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.29.4 Member Data Documentation

32.29.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

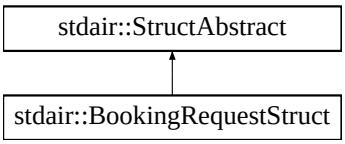
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

32.30 stdair::BookingRequestStruct Struct Reference

Structure holding the elements of a booking request.

`#include <stdair/bom/BookingRequestStruct.hpp>`Inheritance diagram for stdair::BookingRequestStruct::



Public Member Functions

- const DemandGeneratorKey_T & getDemandGeneratorKey () const
- const AirportCode_T & getOrigin () const
- const AirportCode_T & getDestination () const
- const CityCode_T & getPOS () const
- const Date_T & getPreferedDepartureDate () const
- const Duration_T & getPreferredDepartureTime () const
- const DateTime_T & getRequestDateTime () const
- const CabinCode_T & getPreferredCabin () const
- const NbOfSeats_T & getPartySize () const
- const ChannelLabel_T & getBookingChannel () const
- const TripType_T & getTripType () const
- const DayDuration_T & getStayDuration () const
- const FrequentFlyer_T & getFrequentFlyerType () const
- const WTP_T & getWTP () const
- const PriceValue_T & getValueOfTime () const
- const ChangeFees_T & getChangeFees () const
- const Disutility_T & getChangeFeeDisutility () const
- const NonRefundable_T & getNonRefundable () const
- const Disutility_T & getNonRefundableDisutility () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const

- BookingRequestStruct (const DemandGeneratorKey_T &iGeneratorKey, const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime, const ChangeFees_T &iChangeFees, const Disutility_T &iChangeFeeDisutility, const NonRefundable_T &iNonRefundable, const Disutility_T &iNonRefundableDisutility)
- BookingRequestStruct (const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iRequestDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime, const ChangeFees_T &iChangeFees, const Disutility_T &iChangeFeeDisutility, const NonRefundable_T &iNonRefundable, const Disutility_T &iNonRefundableDisutility)
- BookingRequestStruct (const BookingRequestStruct &)
- ~BookingRequestStruct ()

32.30.1 Detailed Description

Structure holding the elements of a booking request.

Definition at line 21 of file BookingRequestStruct.hpp.

32.30.2 Constructor & Destructor Documentation

32.30.2.1 stdair::BookingRequestStruct::BookingRequestStruct (const DemandGeneratorKey_T & *iGeneratorKey*, const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iRequestDateTime*, const CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*, const ChangeFees_T & *iChangeFees*, const Disutility_T & *iChangeFeeDisutility*, const NonRefundable_T & *iNonRefundable*, const Disutility_T & *iNonRefundableDisutility*)

Default constructor.

Definition at line 63 of file BookingRequestStruct.cpp.

32.30.2.2 stdair::BookingRequestStruct::BookingRequestStruct (const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iRequestDateTime*, const CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*, const ChangeFees_T & *iChangeFees*, const Disutility_T & *iChangeFeeDisutility*, const NonRefundable_T & *iNonRefundable*, const Disutility_T & *iNonRefundableDisutility*)

Constructor without the demand generator key, used for batches.

Definition at line 98 of file BookingRequestStruct.cpp.

32.30.2.3 stdair::BookingRequestStruct::BookingRequestStruct (const BookingRequestStruct & *iBookingRequest*)

Copy constructor.

Definition at line 39 of file BookingRequestStruct.cpp.

32.30.2.4 stdair::BookingRequestStruct::~BookingRequestStruct ()

Destructor.

Definition at line 131 of file BookingRequestStruct.cpp.

32.30.3 Member Function Documentation

32.30.3.1 const DemandGeneratorKey_T& stdair::BookingRequestStruct::getDemandGeneratorKey () const [inline]

Get the demand generator key.

Definition at line 25 of file BookingRequestStruct.hpp.

32.30.3.2 const AirportCode_T& stdair::BookingRequestStruct::getOrigin () const [inline]

Get the requested origin.

Definition at line 30 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.3 const AirportCode_T& stdair::BookingRequestStruct::getDestination () const [inline]

Get the requested destination.

Definition at line 35 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.4 const CityCode_T& stdair::BookingRequestStruct::getPOS () const [inline]

Get the point-of-sale.

Definition at line 40 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.5 const Date_T& stdair::BookingRequestStruct::getPreferedDepartureDate () const [inline]

Get the requested departure date.

Definition at line 45 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.6 const Duration_T& stdair::BookingRequestStruct::getPreferredDepartureTime () const [inline]

Get the preferred departure time.

Definition at line 50 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.7 const DateTime_T& stdair::BookingRequestStruct::getRequestDateTime () const [inline]

Get the request datetime.

Definition at line 55 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.8 const CabinCode_T& stdair::BookingRequestStruct::getPreferredCabin () const [inline]

Get the preferred cabin.

Definition at line 60 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.9 const NbOfSeats_T& stdair::BookingRequestStruct::getPartySize () const [inline]

Get the party size.

Definition at line 65 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.10 const ChannelLabel_T& stdair::BookingRequestStruct::getBookingChannel () const [inline]

Get the reservation channel.

Definition at line 70 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.11 const TripType_T& stdair::BookingRequestStruct::getTripType () const [inline]

Get the trip type.

Definition at line 75 of file BookingRequestStruct.hpp.

32.30.3.12 const DayDuration_T& stdair::BookingRequestStruct::getStayDuration () const [inline]

Get the duration of stay.

Definition at line 80 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.13 const FrequentFlyer_T& stdair::BookingRequestStruct::getFrequentFlyerType () const [inline]

Get the frequent flyer type.

Definition at line 85 of file BookingRequestStruct.hpp.

32.30.3.14 const WTP_T& stdair::BookingRequestStruct::getWTP () const [inline]

Get the willingness-to-pay.

Definition at line 90 of file BookingRequestStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.30.3.15 const PriceValue_T& stdair::BookingRequestStruct::getValueOfTime () const [inline]

Get the value of time.

Definition at line 95 of file BookingRequestStruct.hpp.

32.30.3.16 const ChangeFees_T& stdair::BookingRequestStruct::getChangeFees () const [inline]

Get the change fee acceptation.

Definition at line 100 of file BookingRequestStruct.hpp.

32.30.3.17 const Disutility_T& stdair::BookingRequestStruct::getChangeFeeDisutility () const [inline]

Get the change disutility.

Definition at line 105 of file BookingRequestStruct.hpp.

32.30.3.18 const NonRefundable_T& stdair::BookingRequestStruct::getNonRefundable () const [inline]

Get the non refundable acceptation.

Definition at line 110 of file BookingRequestStruct.hpp.

32.30.3.19 const Disutility_T& stdair::BookingRequestStruct::getNonRefundableDisutility () const [inline]

Get the non refundable disutility.

Definition at line 115 of file BookingRequestStruct.hpp.

32.30.3.20 void stdair::BookingRequestStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 135 of file BookingRequestStruct.cpp.

References describe().

32.30.3.21 void stdair::BookingRequestStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 140 of file BookingRequestStruct.cpp.

32.30.3.22 const std::string stdair::BookingRequestStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 144 of file BookingRequestStruct.cpp.

Referenced by toStream().

32.30.3.23 const std::string stdair::BookingRequestStruct::display () const

Display of the structure.

- id,
- request_date (YYMMDD),
- request_time (HHMMSS),
- POS (three-letter code),
- Channel (two-letter code):

  - 'D' for direct or 'I' for indirect,
  - 'N' for oNline or 'F' for oFfline,
- Origin (three-letter code),
- Destination (three-letter code),
- Preferred departure date (YYMMDD),
- Preferred departure time (HHMM),
- Min departure time (HHMM),
- Max departure time (HHMM),
- Preferred arrival date (YYMMDD),
- Preferred arrival time (HHMM),
- Preferred cabin:
  - 'F' for first,
  - 'C' for club/business,
  - 'W' for economy plus,
  - 'M' for economy,
- Trip type:
  - 'OW' for a one-way trip,
  - 'RO' for the outbound part of a rount-trip,
  - 'RI' for the inbound part of a rount-trip,
- Duration of stay (expressed as a number of days),
- Frequent flyer tier:
  - 'G' for gold,
  - 'S' for silver,
  - 'K' for basic,
  - 'N' for none,
- Willingness-to-pay (WTP, expressed as a monetary unit, e.g., EUR),
- Disutility per stop (expressed as a monetary unit, e.g., EUR),
- Value of time (EUR per hour),

Returns:

const std::string The output of the booking request structure.

Definition at line 169 of file BookingRequestStruct.cpp.

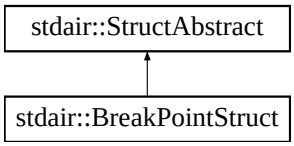
References stdair::TRIP_TYPE_ONE_WAY.

The documentation for this struct was generated from the following files:

- stdair/bom/BookingRequestStruct.hpp
- stdair/bom/BookingRequestStruct.cpp

32.31 stdair::BreakPointStruct Struct Reference

`#include <stdair/bom/BreakPointStruct.hpp>`Inheritance diagram for stdair::BreakPointStruct::



Public Member Functions

- const DateTime_T & getBreakPointTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- BreakPointStruct (const DateTime_T &)
- BreakPointStruct (const Date_T &)
- BreakPointStruct (const BreakPointStruct &)
- ~BreakPointStruct ()

32.31.1 Detailed Description

Structure holding the elements of a break point.

Definition at line 18 of file BreakPointStruct.hpp.

32.31.2 Constructor & Destructor Documentation

32.31.2.1 stdair::BreakPointStruct::BreakPointStruct (const DateTime_T & *iBreakPointTime*)

Constructor.

Definition at line 26 of file BreakPointStruct.cpp.

32.31.2.2 stdair::BreakPointStruct::BreakPointStruct (const Date_T & *iBreakPointDate*)

Constructor.

Definition at line 32 of file BreakPointStruct.cpp.

32.31.2.3 stdair::BreakPointStruct::BreakPointStruct (const BreakPointStruct & *iBreakPoint*)

Copy constructor.

Definition at line 20 of file BreakPointStruct.cpp.

32.31.2.4 stdair::BreakPointStruct::~BreakPointStruct ()

Destructor.

Definition at line 37 of file BreakPointStruct.cpp.

32.31.3 Member Function Documentation

32.31.3.1 const DateTime_T& stdair::BreakPointStruct::getBreakPointTime () const [inline]

Get the break point action time.

Definition at line 22 of file BreakPointStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBreakPointObject().

32.31.3.2 void stdair::BreakPointStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file BreakPointStruct.cpp.

References describe().

32.31.3.3 void stdair::BreakPointStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 46 of file BreakPointStruct.cpp.

32.31.3.4 const std::string stdair::BreakPointStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 50 of file BreakPointStruct.cpp.

Referenced by toStream().

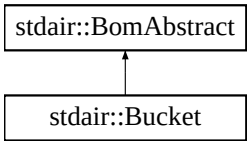
The documentation for this struct was generated from the following files:

- stdair/bom/BreakPointStruct.hpp
- stdair/bom/BreakPointStruct.cpp

32.32 stdair::Bucket Class Reference

Class representing the actual attributes for an airline booking class.

`#include <stdair/bom/Bucket.hpp>`Inheritance diagram for stdair::Bucket::



Public Types

- typedef BucketKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const SeatIndex_T & getSeatIndex () const
- const Yield_T & getYieldRangeUpperValue () const
- const CabinCapacity_T & getAvailability () const
- const NbOfSeats_T & getSoldSeats () const
- void setYieldRangeUpperValue (const Yield_T &iYield)
- void setAvailability (const CabinCapacity_T &iAvl)
- void setSoldSeats (const NbOfSeats_T &iSoldSeats)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Bucket (const Key_T &)
- virtual ~Bucket ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- Yield_T _yieldRangeUpperValue
- CabinCapacity_T _availability
- NbOfSeats_T _soldSeats

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.32.1 Detailed Description

Class representing the actual attributes for an airline booking class.

Definition at line 29 of file Bucket.hpp.

32.32.2 Member Typedef Documentation

32.32.2.1 typedef BucketKey stdair::Bucket::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file Bucket.hpp.

32.32.3 Constructor & Destructor Documentation

32.32.3.1 stdair::Bucket::Bucket (const Key_T & *iKey*) `[protected]`

Default constructor.

Definition at line 34 of file Bucket.cpp.

32.32.3.2 stdair::Bucket::~Bucket () `[protected, virtual]`

Destructor.

Definition at line 38 of file Bucket.cpp.

32.32.4 Member Function Documentation

32.32.4.1 const Key_T& stdair::Bucket::getKey () const `[inline]`

Get the primary key of the bucket.

Definition at line 47 of file Bucket.hpp.

References _key.

32.32.4.2 BomAbstract* const stdair::Bucket::getParent () const `[inline]`

Get the parent object.

Definition at line 54 of file Bucket.hpp.

References _parent.

32.32.4.3 const HolderMap_T& stdair::Bucket::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 59 of file Bucket.hpp.

References _holderMap.

32.32.4.4 const SeatIndex_T& stdair::Bucket::getSeatIndex () const [inline]

Get the seat index (part of the primary key).

Definition at line 64 of file Bucket.hpp.

References _key, and stdair::BucketKey::getSeatIndex().

32.32.4.5 const Yield_T& stdair::Bucket::getYieldRangeUpperValue () const [inline]

Get the upper yield range.

Definition at line 69 of file Bucket.hpp.

References _yieldRangeUpperValue.

32.32.4.6 const CabinCapacity_T& stdair::Bucket::getAvailability () const [inline]

Get the availability.

Definition at line 74 of file Bucket.hpp.

References _availability.

32.32.4.7 const NbOfSeats_T& stdair::Bucket::getSoldSeats () const [inline]

Get the number of seats already sold.

Definition at line 79 of file Bucket.hpp.

References _soldSeats.

32.32.4.8 void stdair::Bucket::setYieldRangeUpperValue (const Yield_T & *iYield*) [inline]

Set the upper yield range.

Definition at line 86 of file Bucket.hpp.

References _yieldRangeUpperValue.

32.32.4.9 void stdair::Bucket::setAvailability (const CabinCapacity_T & *iAvl*) [inline]

Set the availability.

Definition at line 91 of file Bucket.hpp.

References _availability.

32.32.4.10 void stdair::Bucket::setSoldSeats (const NbOfSeats_T & *iSoldSeats*) [inline]

Set the number of seats already sold.

Definition at line 96 of file Bucket.hpp.

References _soldSeats.

32.32.4.11 void stdair::Bucket::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 108 of file Bucket.hpp.

References toString().

32.32.4.12 void stdair::Bucket::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 117 of file Bucket.hpp.

32.32.4.13 std::string stdair::Bucket::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file Bucket.cpp.

References describeKey().

Referenced by toStream().

32.32.4.14 const std::string stdair::Bucket::describeKey () const [inline]

Get a string describing the key.

Definition at line 128 of file Bucket.hpp.

References _key, and stdair::BucketKey::toString().

Referenced by toString().

32.32.4.15 template<class Archive > void stdair::Bucket::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 64 of file Bucket.cpp.

References _key.

32.32.5 Friends And Related Function Documentation

32.32.5.1 friend class FacBom [friend]

Definition at line 30 of file Bucket.hpp.

32.32.5.2 friend class FacCloneBom [friend]

Definition at line 31 of file Bucket.hpp.

32.32.5.3 friend class FacBomManager [friend]

Definition at line 32 of file Bucket.hpp.

32.32.5.4 friend class boost::serialization::access [friend]

Definition at line 33 of file Bucket.hpp.

32.32.6 Member Data Documentation

32.32.6.1 Key_T stdair::Bucket::_key [protected]

Primary key (upper yield range).

Definition at line 179 of file Bucket.hpp.

Referenced by describeKey(), getKey(), getSeatIndex(), and serialize().

32.32.6.2 BomAbstract* stdair::Bucket::_parent [protected]

Pointer on the parent class (LegCabin).

Definition at line 184 of file Bucket.hpp.

Referenced by getParent().

32.32.6.3 HolderMap_T stdair::Bucket::_holderMap [protected]

Map holding the children (empty for now).

Definition at line 189 of file Bucket.hpp.

Referenced by getHolderMap().

32.32.6.4 Yield_T stdair::Bucket::_yieldRangeUpperValue [protected]

Upper yield range.

Definition at line 197 of file Bucket.hpp.

Referenced by getYieldRangeUpperValue(), and setYieldRangeUpperValue().

32.32.6.5 CabinCapacity_T stdair::Bucket::_availability [protected]

Availability.

Definition at line 202 of file Bucket.hpp.

Referenced by getAvailability(), and setAvailability().

32.32.6.6 NbOfSeats_T stdair::Bucket::_soldSeats [protected]

Number of seats already sold.

Definition at line 207 of file Bucket.hpp.

Referenced by getSoldSeats(), and setSoldSeats().

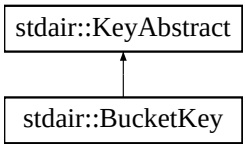
The documentation for this class was generated from the following files:

- stdair/bom/Bucket.hpp
- stdair/bom/Bucket.cpp

32.33 stdair::BucketKey Struct Reference

Key of booking-class.

`#include <stdair/bom/BucketKey.hpp>`Inheritance diagram for stdair::BucketKey::



Public Member Functions

- BucketKey (const SeatIndex_T &)
- BucketKey (const BucketKey &)
- ~BucketKey ()

- const SeatIndex_T & getSeatIndex () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.33.1 Detailed Description

Key of booking-class.

Definition at line 26 of file BucketKey.hpp.

32.33.2 Constructor & Destructor Documentation

32.33.2.1 stdair::BucketKey::BucketKey (const SeatIndex_T & *iSeatIndex*)

Main constructor.

Definition at line 22 of file BucketKey.cpp.

32.33.2.2 stdair::BucketKey::BucketKey (const BucketKey & *iBucketKey*)

Copy constructor.

Definition at line 27 of file BucketKey.cpp.

32.33.2.3 stdair::BucketKey::~BucketKey ()

Destructor.

Definition at line 32 of file BucketKey.cpp.

32.33.3 Member Function Documentation

32.33.3.1 const SeatIndex_T& stdair::BucketKey::getSeatIndex () const `[inline]`

Get the seat index.

Definition at line 54 of file BucketKey.hpp.

Referenced by stdair::Bucket::getSeatIndex().

32.33.3.2 void stdair::BucketKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file BucketKey.cpp.

References toString().

32.33.3.3 void stdair::BucketKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file BucketKey.cpp.

32.33.3.4 const std::string stdair::BucketKey::toString () const `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-cabin.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file BucketKey.cpp.

Referenced by stdair::Bucket::describeKey(), and toStream().

32.33.3.5 template<class Archive > void stdair::BucketKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 67 of file BucketKey.cpp.

32.33.4 Friends And Related Function Documentation

32.33.4.1 friend class boost::serialization::access `[friend]`

Definition at line 27 of file BucketKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/BucketKey.hpp
- stdair/bom/BucketKey.cpp

32.34 stdair_test::Cabin Struct Reference

`#include <test/stdair/StdairTestLib.hpp>`

Public Types

- typedef BookingClass child

Public Member Functions

- Cabin (const BookingClass &iBkgClass)
- std::string toString () const

Public Attributes

- BookingClass _bookingClass

32.34.1 Detailed Description

Cabin

Definition at line 32 of file StdairTestLib.hpp.

32.34.2 Member Typedef Documentation

32.34.2.1 typedef BookingClass stdair_test::Cabin::child

Child type.

Definition at line 46 of file StdairTestLib.hpp.

32.34.3 Constructor & Destructor Documentation

32.34.3.1 stdair_test::Cabin::Cabin (const BookingClass & *iBkgClass*) `[inline]`

Definition at line 34 of file StdairTestLib.hpp.

32.34.4 Member Function Documentation

32.34.4.1 std::string stdair_test::Cabin::toString () const `[inline]`

Display .

Definition at line 39 of file StdairTestLib.hpp.

References _bookingClass, and stdair_test::BookingClass::_classCode.

32.34.5 Member Data Documentation

32.34.5.1 BookingClass stdair_test::Cabin::_bookingClass

Definition at line 33 of file StdairTestLib.hpp.

Referenced by toString().

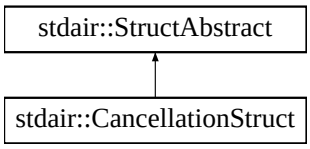
The documentation for this struct was generated from the following file:

- test/stdair/StdairTestLib.hpp

32.35 stdair::CancellationStruct Struct Reference

Structure holding the elements of a travel solution.

`#include <stdair/bom/CancellationStruct.hpp>`Inheritance diagram for stdair::CancellationStruct::



Public Member Functions

- const SegmentPath_T & getSegmentPath () const
- const ClassList_String_T & getClassList () const
- const BookingClassIDList_T & getClassIDList () const
- const PartySize_T & getPartySize () const
- const DateTime_T & getCancellationDateTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const
- CancellationStruct (const SegmentPath_T &, const ClassList_String_T &, const PartySize_T &, const DateTime_T &)
- CancellationStruct (const SegmentPath_T &, const BookingClassIDList_T &, const PartySize_T &, const DateTime_T &)
- ~CancellationStruct ()

32.35.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 23 of file CancellationStruct.hpp.

32.35.2 Constructor & Destructor Documentation

32.35.2.1 stdair::CancellationStruct::CancellationStruct (const SegmentPath_T & *iSegPath*, const ClassList_String_T & *iList*, const PartySize_T & *iSize*, const DateTime_T & *iDateTime*)

Default constructor without class ID list.

Definition at line 14 of file CancellationStruct.cpp.

32.35.2.2 stdair::CancellationStruct::CancellationStruct (const SegmentPath_T & *iSegPath*, const BookingClassIDList_T & *iIDList*, const PartySize_T & *iSize*, const DateTime_T & *iDateTime*)

Default constructor with class ID list.

Definition at line 23 of file CancellationStruct.cpp.

32.35.2.3 stdair::CancellationStruct::~CancellationStruct ()

Destructor.

Definition at line 32 of file CancellationStruct.cpp.

32.35.3 Member Function Documentation

32.35.3.1 const SegmentPath_T& stdair::CancellationStruct::getSegmentPath () const [inline]

Get the segment path.

Definition at line 27 of file CancellationStruct.hpp.

32.35.3.2 const ClassList_String_T& stdair::CancellationStruct::getClassList () const [inline]

Get the class list.

Definition at line 32 of file CancellationStruct.hpp.

32.35.3.3 const BookingClassIDList_T& stdair::CancellationStruct::getClassIDList () const [inline]

Get the class ID list.

Definition at line 37 of file CancellationStruct.hpp.

32.35.3.4 const PartySize_T& stdair::CancellationStruct::getPartySize () const [inline]

Get the party size.

Definition at line 42 of file CancellationStruct.hpp.

32.35.3.5 const DateTime_T& stdair::CancellationStruct::getCancellationDateTime () const [inline]

Get the datetime.

Definition at line 47 of file CancellationStruct.hpp.

32.35.3.6 void stdair::CancellationStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file CancellationStruct.cpp.

References describe().

32.35.3.7 void stdair::CancellationStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file CancellationStruct.cpp.

32.35.3.8 const std::string stdair::CancellationStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file CancellationStruct.cpp.

References stdair::BookingClass::getClassCode(), and stdair::BomID< BOM >::getObject().

Referenced by toStream().

32.35.3.9 const std::string stdair::CancellationStruct::display () const

Display of the structure.

Definition at line 81 of file CancellationStruct.cpp.

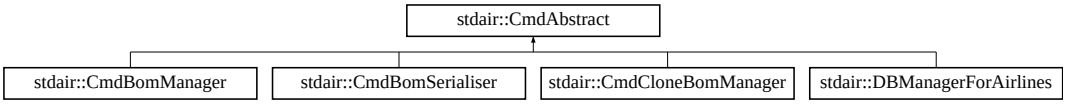
References stdair::BookingClass::getClassCode(), and stdair::BomID< BOM >::getObject().

The documentation for this struct was generated from the following files:

- stdair/bom/CancellationStruct.hpp
- stdair/bom/CancellationStruct.cpp

32.36 stdair::CmdAbstract Class Reference

`#include <stdair/command/CmdAbstract.hpp>`Inheritance diagram for stdair::CmdAbstract::



32.36.1 Detailed Description

Base class for the Command layer.

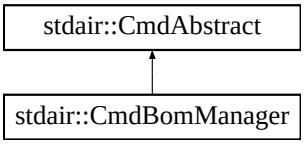
Definition at line 11 of file CmdAbstract.hpp.

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

- stdair/command/CmdAbstract.hpp

32.37 stdair::CmdBomManager Class Reference

`#include <stdair/command/CmdBomManager.hpp>`Inheritance diagram for stdair::CmdBomManager::



Friends

- class STDAIR_Service

32.37.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 25 of file CmdBomManager.hpp.

32.37.2 Friends And Related Function Documentation

32.37.2.1 friend class STDAIR_Service `[friend]`

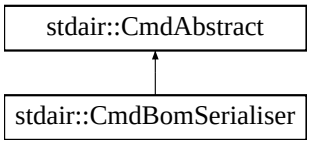
Definition at line 27 of file CmdBomManager.hpp.

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

- stdair/command/CmdBomManager.hpp

32.38 stdair::CmdBomSerialiser Class Reference

`#include <stdair/command/CmdBomSerialiser.hpp>`Inheritance diagram for stdair::CmdBomSerialiser::



32.38.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

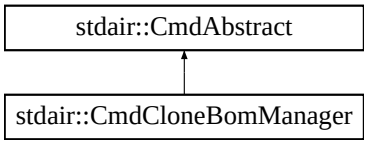
Definition at line 25 of file CmdBomSerialiser.hpp.

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

- stdair/command/CmdBomSerialiser.hpp

32.39 stdair::CmdCloneBomManager Class Reference

`#include <stdair/command/CmdCloneBomManager.hpp>`Inheritance diagram for stdair::CmdCloneBomManager::



Friends

- class STDAIR_Service

32.39.1 Detailed Description

Class wrapping utility functions for handling the BOM tree objects.

Definition at line 40 of file CmdCloneBomManager.hpp.

32.39.2 Friends And Related Function Documentation

32.39.2.1 friend class STDAIR_Service `[friend]`

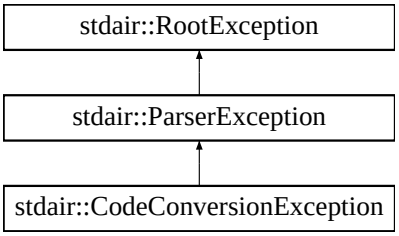
Definition at line 42 of file CmdCloneBomManager.hpp.

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

- stdair/command/CmdCloneBomManager.hpp

32.40 stdair::CodeConversionException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::CodeConversionException::



Public Member Functions

- CodeConversionException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.40.1 Detailed Description

Code conversion.

Definition at line 133 of file stdair_exceptions.hpp.

32.40.2 Constructor & Destructor Documentation

32.40.2.1 stdair::CodeConversionException::CodeConversionException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 136 of file stdair_exceptions.hpp.

32.40.3 Member Function Documentation

32.40.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.40.4 Member Data Documentation

32.40.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

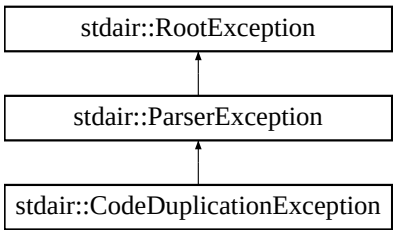
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.41 stdair::CodeDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::CodeDuplicationException::



Public Member Functions

- CodeDuplicationException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.41.1 Detailed Description

Code duplication.

Definition at line 141 of file stdair_exceptions.hpp.

32.41.2 Constructor & Destructor Documentation

32.41.2.1 stdair::CodeDuplicationException::CodeDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 144 of file stdair_exceptions.hpp.

32.41.3 Member Function Documentation

32.41.3.1 const char∗ stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.41.4 Member Data Documentation

32.41.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.42 COMMAND Struct Reference

```
#include <stdair/ui/cmdline/readline_autocomp.hpp>
```

Public Attributes

- char const ∗ name
- pt2Func ∗ func
- char ∗ doc

32.42.1 Detailed Description

A structure which contains information on the commands this program can understand.

Definition at line 41 of file readline_autocomp.hpp.

32.42.2 Member Data Documentation

32.42.2.1 char const∗ COMMAND::name

User printable name of the function.

Definition at line 45 of file readline_autocomp.hpp.

Referenced by com_help(), and find_command().

32.42.2.2 pt2Func* COMMAND::func

Function to call to do the job.

Definition at line 50 of file readline_autocomp.hpp.

Referenced by execute_line().

32.42.2.3 char* COMMAND::doc

Documentation for this function.

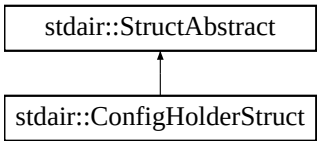
Definition at line 55 of file readline_autocomp.hpp.

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

- stdair/ui/cmdline/readline_autocomp.hpp

32.43 stdair::ConfigHolderStruct Struct Reference

`#include <stdair/bom/ConfigHolderStruct.hpp>`Inheritance diagram for stdair::ConfigHolderStruct::



Public Member Functions

- void add (const bpt::ptree &)
- bool addValue (const std::string &iValue, const std::string &iPath)
- template<typename ValueType >
  bool exportValue (ValueType &ioValue, const std::string &iPath) const
- void updateAirlineFeatures (BomRoot &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string jsonExport () const
- ConfigHolderStruct ()
- ConfigHolderStruct (const ConfigHolderStruct &)
- ~ConfigHolderStruct ()
- template<>
  bool exportValue (Date_T &ioValue, const std::string &iPath) const

32.43.1 Detailed Description

Structure holding the configuration of the simulation.

Definition at line 40 of file ConfigHolderStruct.hpp.

32.43.2 Constructor & Destructor Documentation

32.43.2.1 stdair::ConfigHolderStruct::ConfigHolderStruct ()

Constructor.

Definition at line 27 of file ConfigHolderStruct.cpp.

32.43.2.2 stdair::ConfigHolderStruct::ConfigHolderStruct (const ConfigHolderStruct & *iConfigHolderStruct*)

Copy constructor.

Definition at line 32 of file ConfigHolderStruct.cpp.

32.43.2.3 stdair::ConfigHolderStruct::~ConfigHolderStruct ()

Destructor.

Definition at line 37 of file ConfigHolderStruct.cpp.

32.43.3 Member Function Documentation

32.43.3.1 void stdair::ConfigHolderStruct::add (const bpt::ptree & *iConfigPropertyTree*)

Merge the given property tree with the existing configuration property tree gathering all the configuration information.

Parameters:

const bpt::ptree& Property tree to add to the configuration tree.

Definition at line 144 of file ConfigHolderStruct.cpp.

Referenced by stdair::BomINIImport::importINIConfig(), and stdair::BomJSONImport::jsonImportConfig().

32.43.3.2 bool stdair::ConfigHolderStruct::addValue (const std::string & *iValue*, const std::string & *iPath*)

Create the given specified path in the configuration tree and add the corresponding given value (or replace the value if the path already exists).

Parameters:

const std::string& Value to add at the given path.

const std::string& Path to create (or to look for).

Definition at line 191 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::importConfigValue().

32.43.3.3 template<typename ValueType > bool stdair::ConfigHolderStruct::exportValue (ValueType & *ioValue*, const std::string & *iPath*) const [inline]

Look for the specified path in the configuration tree and, if existing, try to extract the corresponding value. The type of the value to extract is a template parameter.

Parameters:

ValueType& Value to add in the configuration tree.

const std::string& Path to look for.

Definition at line 144 of file ConfigHolderStruct.hpp.

Referenced by stdair::STDAIR_Service::exportConfigValue().

32.43.3.4 void stdair::ConfigHolderStruct::updateAirlineFeatures (BomRoot & *iBomRoot*)

Update the airline features objects thanks to the configuration holder.

Parameters:

BomRoot& Reference on the BomRoot to update.

Definition at line 220 of file ConfigHolderStruct.cpp.

References stdair::BomRetriever::retrieveAirlineFeatureFromKey(), stdair::AirlineFeature::setForecastingMethod(), stdair::AirlineFeature::setOptimisationMethod(), stdair::AirlineFeature::setPartnershipTechnique(), stdair::AirlineFeature::setPreOptimisationMethod(), stdair::AirlineFeature::setUnconstrainingMethod(), STDAIR_LOG_ERROR, and stdair::RootException::what().

Referenced by stdair::STDAIR_Service::updateAirlineFeatures().

32.43.3.5 void stdair::ConfigHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file ConfigHolderStruct.cpp.

References describe().

32.43.3.6 void stdair::ConfigHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 46 of file ConfigHolderStruct.cpp.

32.43.3.7 const std::string stdair::ConfigHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 50 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::configDisplay(), and toStream().

32.43.3.8 const std::string stdair::ConfigHolderStruct::jsonExport () const

Display of the configuration in a JSON-ified format.

Definition at line 134 of file ConfigHolderStruct.cpp.

Referenced by stdair::STDAIR_Service::jsonExportConfiguration().

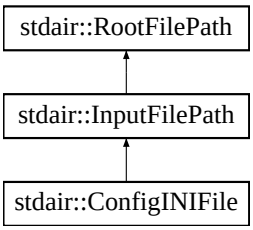
32.43.3.9 template<> bool stdair::ConfigHolderStruct::exportValue (Date_T & *ioValue*, const std::string & *iPath*) const [inline]

The documentation for this struct was generated from the following files:

- stdair/bom/ConfigHolderStruct.hpp
- stdair/bom/ConfigHolderStruct.cpp

32.44 stdair::ConfigINIFile Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::ConfigINIFile::



Public Member Functions

- ConfigINIFile (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.44.1 Detailed Description

Config file: INI format

Definition at line 112 of file stdair_file.hpp.

32.44.2 Constructor & Destructor Documentation

32.44.2.1 stdair::ConfigINIFile::ConfigINIFile (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 117 of file stdair_file.hpp.

32.44.3 Member Function Documentation

32.44.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.44.4 Member Data Documentation

32.44.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

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

- stdair/stdair_file.hpp

32.45 stdair::ContinuousAttributeLite< T > Struct Template Reference

Class modeling the distribution of values that can be taken by a continuous attribute.

```
#include <stdair/basic/ContinuousAttributeLite.hpp>
```

Public Types

- typedef std::map< T, stdair::Probability_T > ContinuousDistribution_T

Public Member Functions

- const T getValue (const stdair::Probability_T &iCumulativeProbability) const

- const stdair::Probability_T getRemainingProportion (const T &iValue) const
- const double getDerivativeValue (const T iKey) const
- const T getUpperBound (const T iKey) const
- const std::string displayCumulativeDistribution () const
- ContinuousAttributeLite (const ContinuousDistribution_T &iValueMap)
- ContinuousAttributeLite (const ContinuousAttributeLite &iCAL)
- ContinuousAttributeLite & operator= (const ContinuousAttributeLite &iCAL)
- virtual ~ContinuousAttributeLite ()

32.45.1 Detailed Description

template<typename T> struct stdair::ContinuousAttributeLite< T >

Class modeling the distribution of values that can be taken by a continuous attribute.

Definition at line 26 of file ContinuousAttributeLite.hpp.

32.45.2 Member Typedef Documentation

32.45.2.1 template<typename T > typedef std::map<T, stdair::Probability_T> stdair::ContinuousAttributeLite< T >::ContinuousDistribution_T

Type for the probability mass function.

Definition at line 32 of file ContinuousAttributeLite.hpp.

32.45.3 Constructor & Destructor Documentation

32.45.3.1 template<typename T > stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite (const ContinuousDistribution_T & *iValueMap*) `[inline]`

Constructor.

Definition at line 204 of file ContinuousAttributeLite.hpp.

32.45.3.2 template<typename T > stdair::ContinuousAttributeLite< T >::ContinuousAttributeLite (const ContinuousAttributeLite< T > & *iCAL*) `[inline]`

Copy constructor.

Definition at line 212 of file ContinuousAttributeLite.hpp.

32.45.3.3 template<typename T > virtual stdair::ContinuousAttributeLite< T >::~ContinuousAttributeLite () `[inline, virtual]`

Destructor.

Definition at line 231 of file ContinuousAttributeLite.hpp.

32.45.4 Member Function Documentation

32.45.4.1 template<typename T > const T stdair::ContinuousAttributeLite< T >::getValue (const stdair::Probability_T & *iCumulativeProbability*) const [inline]

Get value from inverse cumulative distribution.

Definition at line 39 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue(), and stdair::DictionaryManager::valueToKey().

32.45.4.2 template<typename T > const stdair::Probability_T stdair::ContinuousAttributeLite< T >::getRemainingProportion (const T & *iValue*) const [inline]

Get remaining proportion from cumulative distribution.

Definition at line 84 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

32.45.4.3 template<typename T > const double stdair::ContinuousAttributeLite< T >::getDerivativeValue (const T *iKey*) const [inline]

Get the value of the derivative function in a key point.

Definition at line 131 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

32.45.4.4 template<typename T > const T stdair::ContinuousAttributeLite< T >::getUpperBound (const T *iKey*) const [inline]

Get the upper bound.

Definition at line 163 of file ContinuousAttributeLite.hpp.

32.45.4.5 template<typename T > const std::string stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution () const [inline]

Display cumulative distribution.

Definition at line 182 of file ContinuousAttributeLite.hpp.

References stdair::DictionaryManager::keyToValue().

32.45.4.6 template<typename T > ContinuousAttributeLite& stdair::ContinuousAttributeLite< T >::operator= (const ContinuousAttributeLite< T > & *iCAL*) [inline]

Copy operator.

Definition at line 221 of file ContinuousAttributeLite.hpp.

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

- stdair/basic/ContinuousAttributeLite.hpp

32.46 stdair::date_time_element< MIN, MAX > Struct Template Reference

```
#include <stdair/basic/BasParserHelperTypes.hpp>
```

Public Member Functions

- date_time_element ()
- date_time_element (const date_time_element &t)
- date_time_element (int i)
- void check () const

Public Attributes

- unsigned int _value

32.46.1 Detailed Description

template<int MIN = 0, int MAX = 0> struct stdair::date_time_element< MIN, MAX >

Date & time element parser.

Definition at line 23 of file BasParserHelperTypes.hpp.

32.46.2 Constructor & Destructor Documentation

32.46.2.1 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element () [inline]

Default constructor.

Definition at line 28 of file BasParserHelperTypes.hpp.

32.46.2.2 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (const date_time_element< MIN, MAX > & *t*) [inline]

Default copy constructor.

Definition at line 30 of file BasParserHelperTypes.hpp.

32.46.2.3 template<int MIN = 0, int MAX = 0> stdair::date_time_element< MIN, MAX >::date_time_element (int *i*) [inline]

Constructor.

Definition at line 32 of file BasParserHelperTypes.hpp.

32.46.3 Member Function Documentation

32.46.3.1 template<int MIN = 0, int MAX = 0> void stdair::date_time_element< MIN, MAX >::check () const [inline]

Checker.

Definition at line 34 of file BasParserHelperTypes.hpp.

References stdair::date_time_element< MIN, MAX >::_value.

32.46.4 Member Data Documentation

32.46.4.1 template<int MIN = 0, int MAX = 0> unsigned int stdair::date_time_element< MIN, MAX >::_value

Definition at line 24 of file BasParserHelperTypes.hpp.

Referenced by stdair::date_time_element< MIN, MAX >::check(), stdair::operator*(), and stdair::operator+().

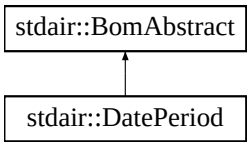
The documentation for this struct was generated from the following file:

- stdair/basic/BasParserHelperTypes.hpp

32.47 stdair::DatePeriod Class Reference

Class representing the actual attributes for a fare date-period.

`#include <stdair/bom/DatePeriod.hpp>`Inheritance diagram for stdair::DatePeriod::



Public Types

- typedef DatePeriodKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const DatePeriod_T & getDatePeriod () const
- bool isDepartureDateValid (const Date_T &) const

Protected Member Functions

- DatePeriod (const Key_T &)
- virtual ~DatePeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.47.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file DatePeriod.hpp.

32.47.2 Member Typedef Documentation

32.47.2.1 typedef DatePeriodKey stdair::DatePeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file DatePeriod.hpp.

32.47.3 Constructor & Destructor Documentation

32.47.3.1 stdair::DatePeriod::DatePeriod (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 27 of file DatePeriod.cpp.

32.47.3.2 stdair::DatePeriod::~DatePeriod () `[protected, virtual]`

Destructor.

Definition at line 32 of file DatePeriod.cpp.

32.47.4 Member Function Documentation

32.47.4.1 void stdair::DatePeriod::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file DatePeriod.hpp.

References toString().

32.47.4.2 void stdair::DatePeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file DatePeriod.hpp.

32.47.4.3 std::string stdair::DatePeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file DatePeriod.cpp.

References describeKey().

Referenced by toStream().

32.47.4.4 const std::string stdair::DatePeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file DatePeriod.hpp.

References _key, and stdair::DatePeriodKey::toString().

Referenced by toString().

32.47.4.5 const Key_T& stdair::DatePeriod::getKey () const [inline]

Get the primary key (date period).

Definition at line 66 of file DatePeriod.hpp.

References _key.

32.47.4.6 BomAbstract* const stdair::DatePeriod::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 73 of file DatePeriod.hpp.

References _parent.

32.47.4.7 const HolderMap_T& stdair::DatePeriod::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 80 of file DatePeriod.hpp.

References _holderMap.

32.47.4.8 const DatePeriod_T& stdair::DatePeriod::getDatePeriod () const [inline]

Get the date period.

Definition at line 87 of file DatePeriod.hpp.

References _key, and stdair::DatePeriodKey::getDatePeriod().

Referenced by isDepartureDateValid().

32.47.4.9 bool stdair::DatePeriod::isDepartureDateValid (const Date_T & *iFlightDate*) const

Check if the given departure date is included in the departure period of the segment path.

Definition at line 44 of file DatePeriod.cpp.

References getDatePeriod().

Referenced by stdair::BomRetriever::retrieveDatePeriodListFromKey().

32.47.5 Friends And Related Function Documentation

32.47.5.1 friend class FacBom [friend]

Definition at line 19 of file DatePeriod.hpp.

32.47.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file DatePeriod.hpp.

32.47.5.3 friend class FacBomManager [friend]

Definition at line 21 of file DatePeriod.hpp.

32.47.6 Member Data Documentation

32.47.6.1 Key_T stdair::DatePeriod::_key [protected]

Primary key (date period).

Definition at line 126 of file DatePeriod.hpp.

Referenced by describeKey(), getDatePeriod(), and getKey().

32.47.6.2 BomAbstract* stdair::DatePeriod::_parent [protected]

Pointer on the parent class.

Definition at line 131 of file DatePeriod.hpp.

Referenced by getParent().

32.47.6.3 HolderMap_T stdair::DatePeriod::_holderMap [protected]

Map holding the children.

Definition at line 136 of file DatePeriod.hpp.

Referenced by getHolderMap().

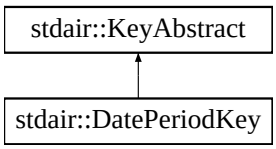
The documentation for this class was generated from the following files:

- stdair/bom/DatePeriod.hpp
- stdair/bom/DatePeriod.cpp

32.48 stdair::DatePeriodKey Struct Reference

Key of date-period.

`#include <stdair/bom/DatePeriodKey.hpp>`Inheritance diagram for stdair::DatePeriodKey::



Public Member Functions

- DatePeriodKey (const DatePeriod_T &)
- DatePeriodKey (const DatePeriodKey &)
- ~DatePeriodKey ()
- const DatePeriod_T & getDatePeriod () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.48.1 Detailed Description

Key of date-period.

Definition at line 14 of file DatePeriodKey.hpp.

32.48.2 Constructor & Destructor Documentation

32.48.2.1 stdair::DatePeriodKey::DatePeriodKey (const DatePeriod_T & *iDatePeriod*)

Main Constructor.

Definition at line 22 of file DatePeriodKey.cpp.

32.48.2.2 stdair::DatePeriodKey::DatePeriodKey (const DatePeriodKey & *iKey*)

Copy constructor.

Definition at line 27 of file DatePeriodKey.cpp.

32.48.2.3 stdair::DatePeriodKey::~DatePeriodKey ()

Destructor.

Definition at line 32 of file DatePeriodKey.cpp.

32.48.3 Member Function Documentation

32.48.3.1 const DatePeriod_T& stdair::DatePeriodKey::getDatePeriod () const `[inline]`

Get the date period.

Definition at line 32 of file DatePeriodKey.hpp.

Referenced by stdair::DatePeriod::getDatePeriod().

32.48.3.2 void stdair::DatePeriodKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file DatePeriodKey.cpp.

References toString().

32.48.3.3 void stdair::DatePeriodKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file DatePeriodKey.cpp.

32.48.3.4 const std::string stdair::DatePeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file DatePeriodKey.cpp.

Referenced by stdair::DatePeriod::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/DatePeriodKey.hpp
- stdair/bom/DatePeriodKey.cpp

32.49 stdair::DbaAbstract Class Reference

```
#include <stdair/dbadaptor/DbaAbstract.hpp>
```

Public Member Functions

- virtual ~DbaAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- DbaAbstract ()

32.49.1 Detailed Description

Base class for the Database Adaptor (DBA) layer.

Definition at line 13 of file DbaAbstract.hpp.

32.49.2 Constructor & Destructor Documentation

32.49.2.1 virtual stdair::DbaAbstract::~DbaAbstract () [inline, virtual]

Destructor.

Definition at line 17 of file DbaAbstract.hpp.

32.49.2.2 stdair::DbaAbstract::DbaAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 29 of file DbaAbstract.hpp.

32.49.3 Member Function Documentation

32.49.3.1 virtual void stdair::DbaAbstract::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 21 of file DbaAbstract.hpp.

32.49.3.2 virtual void stdair::DbaAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 25 of file DbaAbstract.hpp.

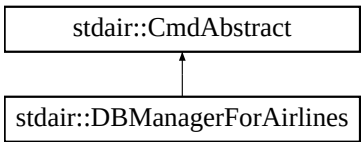
Referenced by operator>>().

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

- stdair/dbadaptor/DbaAbstract.hpp

32.50 stdair::DBManagerForAirlines Class Reference

`#include <stdair/command/DBManagerForAirlines.hpp>`Inheritance diagram for stdair::DBManagerForAirlines::



Static Public Member Functions

- static void updateAirlineInDB (DBSession_T &, const AirlineStruct &)
- static bool retrieveAirline (DBSession_T &, const AirlineCode_T &, AirlineStruct &)
- static void prepareSelectStatement (DBSession_T &, DBRequestStatement_T &, AirlineStruct &)
- static bool iterateOnStatement (DBRequestStatement_T &, AirlineStruct &)

32.50.1 Detailed Description

Class building the Business Object Model (BOM) from data retrieved from the database.

Definition at line 18 of file DBManagerForAirlines.hpp.

32.50.2 Member Function Documentation

32.50.2.1 void stdair::DBManagerForAirlines::updateAirlineInDB (DBSession_T & *ioSociSession*, const AirlineStruct & *iAirline*) [static]

Update the fields of the database row corresponding to the given BOM object. DBSession_T& AirlineStruct& .

Definition at line 99 of file DBManagerForAirlines.cpp.

References stdair::AirlineStruct::getAirlineCode(), and stdair::AirlineStruct::getAirlineName().

32.50.2.2 bool stdair::DBManagerForAirlines::retrieveAirline (DBSession_T & *ioSociSession*, const AirlineCode_T & *iAirlineCode*, AirlineStruct & *ioAirline*) [static]

Retrieve, from the (MySQL) database, the row corresponding to the given BOM code, and fill the given BOM object with that retrieved data. DBSession_T& const AirlineCode_T& AirlineStruct& .

Definition at line 134 of file DBManagerForAirlines.cpp.

References iterateOnStatement().

32.50.2.3 void stdair::DBManagerForAirlines::prepareSelectStatement (DBSession_T & *ioSociSession*, DBRequestStatement_T & *ioSelectStatement*, AirlineStruct & *ioAirline*) [static]

Prepare (parse and put in cache) the SQL statement. DBSession_T& DBRequestStatement_T& AirlineStruct& .

Definition at line 26 of file DBManagerForAirlines.cpp.

32.50.2.4 bool stdair::DBManagerForAirlines::iterateOnStatement (DBRequestStatement_T & *ioStatement*, AirlineStruct & *ioAirline*) [static]

Iterate on the SQL statement.

The SQL has to be already prepared. DBRequestStatement_T& AirlineStruct& .

Definition at line 82 of file DBManagerForAirlines.cpp.

Referenced by retrieveAirline().

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

- stdair/command/DBManagerForAirlines.hpp
- stdair/command/DBManagerForAirlines.cpp

32.51 stdair::DBSessionManager Class Reference

```
#include <stdair/service/DBSessionManager.hpp>
```

Public Member Functions

- DBSession_T & getDBSession () const

Static Public Member Functions

- static DBSessionManager & instance ()

Friends

- class FacSupervisor
- class STDAIR_Service

32.51.1 Detailed Description

Class holding the database session.

Note that the database access is handled by the SOCI library.

Definition at line 17 of file DBSessionManager.hpp.

32.51.2 Member Function Documentation

32.51.2.1 DBSessionManager & stdair::DBSessionManager::instance () `[static]`

Return the static DBSessionManager instance.

Definition at line 82 of file DBSessionManager.cpp.

32.51.2.2 DBSession_T & stdair::DBSessionManager::getDBSession () const

Retrieve the database session handler, held by the static instance of DBSessionManager.

Definition at line 92 of file DBSessionManager.cpp.

32.51.3 Friends And Related Function Documentation

32.51.3.1 friend class FacSupervisor `[friend]`

Definition at line 19 of file DBSessionManager.hpp.

32.51.3.2 friend class STDAIR_Service `[friend]`

Definition at line 20 of file DBSessionManager.hpp.

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

- stdair/service/DBSessionManager.hpp
- stdair/service/DBSessionManager.cpp

32.52 stdair::DefaultDCPList Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static DCPList_T init ()

32.52.1 Detailed Description

Definition at line 126 of file BasConst_Inventory.hpp.

32.52.2 Member Function Documentation

32.52.2.1 DCPList_T stdair::DefaultDCPList::init () [static]

Definition at line 518 of file BasConst.cpp.

The documentation for this struct was generated from the following files:

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

32.53 stdair::DefaultDtdFratMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static DTDFratMap_T init ()

32.53.1 Detailed Description

Definition at line 130 of file BasConst_Inventory.hpp.

32.53.2 Member Function Documentation

32.53.2.1 DTDFratMap_T stdair::DefaultDtdFratMap::init () [static]

Definition at line 697 of file BasConst.cpp.

The documentation for this struct was generated from the following files:

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

32.54 stdair::DefaultDtdProbMap Struct Reference

```
#include <stdair/basic/BasConst_Inventory.hpp>
```

Static Public Member Functions

- static DTDProbMap_T init ()

32.54.1 Detailed Description

Definition at line 134 of file BasConst_Inventory.hpp.

32.54.2 Member Function Documentation

32.54.2.1 DTDProbMap_T stdair::DefaultDtdProbMap::init () [static]

Definition at line 714 of file BasConst.cpp.

The documentation for this struct was generated from the following files:

- stdair/basic/BasConst_Inventory.hpp
- stdair/basic/BasConst.cpp

32.55 stdair::DefaultMap Struct Reference

```
#include <stdair/basic/BasConst_SellUpCurves.hpp>
```

Static Public Member Functions

- static FRAT5Curve_T createFRAT5CurveA ()
- static FRAT5Curve_T createFRAT5CurveB ()
- static FRAT5Curve_T createFRAT5CurveC ()
- static FRAT5Curve_T createFRAT5CurveD ()
- static FFDisutilityCurve_T createFFDisutilityCurveA ()
- static FFDisutilityCurve_T createFFDisutilityCurveB ()
- static FFDisutilityCurve_T createFFDisutilityCurveC ()
- static FFDisutilityCurve_T createFFDisutilityCurveD ()
- static FFDisutilityCurve_T createFFDisutilityCurveE ()
- static FFDisutilityCurve_T createFFDisutilityCurveF ()

32.55.1 Detailed Description

FRAT5 curves.

Definition at line 27 of file BasConst_SellUpCurves.hpp.

32.55.2 Member Function Documentation

32.55.2.1 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveA () [static]

Definition at line 533 of file BasConst.cpp.

32.55.2.2 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveB () [static]

Definition at line 547 of file BasConst.cpp.

32.55.2.3 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveC () [static]

Definition at line 561 of file BasConst.cpp.

32.55.2.4 FRAT5Curve_T stdair::DefaultMap::createFRAT5CurveD () [static]

Definition at line 575 of file BasConst.cpp.

32.55.2.5 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveA () [static]

Definition at line 593 of file BasConst.cpp.

32.55.2.6 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveB () [static]

Definition at line 611 of file BasConst.cpp.

32.55.2.7 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveC () [static]

Definition at line 629 of file BasConst.cpp.

32.55.2.8 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveD () [static]

Definition at line 647 of file BasConst.cpp.

32.55.2.9 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveE () [static]

Definition at line 665 of file BasConst.cpp.

32.55.2.10 FFDisutilityCurve_T stdair::DefaultMap::createFFDisutilityCurveF () [static]

Definition at line 683 of file BasConst.cpp.

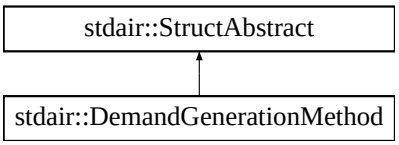
The documentation for this struct was generated from the following files:

- stdair/basic/BasConst_SellUpCurves.hpp
- stdair/basic/BasConst.cpp

32.56 stdair::DemandGenerationMethod Struct Reference

Enumeration of demand (booking request) generation methods.

`#include <stdair/basic/DemandGenerationMethod.hpp>`Inheritance diagram for stdair::DemandGenerationMethod::



Public Types

- enum EN_DemandGenerationMethod { POI_PRO = 0, STA_ORD, LAST_VALUE }

Public Member Functions

- EN_DemandGenerationMethod getMethod () const
- char getMethodAsChar () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_DemandGenerationMethod &) const
- DemandGenerationMethod (const EN_DemandGenerationMethod &)
- DemandGenerationMethod (const char iMethod)
- DemandGenerationMethod (const std::string &iMethod)
- DemandGenerationMethod (const DemandGenerationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_DemandGenerationMethod &)
- static EN_DemandGenerationMethod getMethod (const char)
- static char getMethodLabel (const EN_DemandGenerationMethod &)
- static std::string getMethodLabelAsString (const EN_DemandGenerationMethod &)
- static std::string describeLabels ()

32.56.1 Detailed Description

Enumeration of demand (booking request) generation methods.

Definition at line 17 of file DemandGenerationMethod.hpp.

32.56.2 Member Enumeration Documentation

32.56.2.1 enum stdair::DemandGenerationMethod::EN_DemandGenerationMethod

Enumerator:

POI_PRO

STA_ORD

LAST_VALUE

Definition at line 19 of file DemandGenerationMethod.hpp.

32.56.3 Constructor & Destructor Documentation

32.56.3.1 stdair::DemandGenerationMethod::DemandGenerationMethod (const EN_DemandGenerationMethod & *iDemandGenerationMethod*)

Main constructor.

Definition at line 34 of file DemandGenerationMethod.cpp.

32.56.3.2 stdair::DemandGenerationMethod::DemandGenerationMethod (const char *iMethod*)

Alternative constructor.

Definition at line 62 of file DemandGenerationMethod.cpp.

32.56.3.3 stdair::DemandGenerationMethod::DemandGenerationMethod (const std::string & *iMethod*)

Alternative constructor.

Definition at line 68 of file DemandGenerationMethod.cpp.

References getMethod().

32.56.3.4 stdair::DemandGenerationMethod::DemandGenerationMethod (const DemandGenerationMethod & *iDemandGenerationMethod*)

Default copy constructor.

Definition at line 28 of file DemandGenerationMethod.cpp.

32.56.4 Member Function Documentation

32.56.4.1 const std::string & stdair::DemandGenerationMethod::getLabel (const EN_DemandGenerationMethod & *iMethod*) `[static]`

Get the label as a string (e.g., "PoissonProcess" or "StatisticsOrder").

Definition at line 78 of file DemandGenerationMethod.cpp.

32.56.4.2 DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::getMethod (const char *iMethodChar*) [static]

Get the method value from parsing a single char (e.g., 'P' or 'S').

Definition at line 40 of file DemandGenerationMethod.cpp.

References describeLabels(), LAST_VALUE, POI_PRO, and STA_ORD.

32.56.4.3 char stdair::DemandGenerationMethod::getMethodLabel (const EN_DemandGenerationMethod & *iMethod*) [static]

Get the label as a single char (e.g., 'P' or 'S').

Definition at line 84 of file DemandGenerationMethod.cpp.

32.56.4.4 std::string stdair::DemandGenerationMethod::getMethodLabelAsString (const EN_DemandGenerationMethod & *iMethod*) [static]

Get the label as a string of a single char (e.g., "P" or "S").

Definition at line 90 of file DemandGenerationMethod.cpp.

32.56.4.5 std::string stdair::DemandGenerationMethod::describeLabels () [static]

List the labels.

Definition at line 97 of file DemandGenerationMethod.cpp.

References LAST_VALUE.

Referenced by getMethod().

32.56.4.6 DemandGenerationMethod::EN_DemandGenerationMethod stdair::DemandGenerationMethod::getMethod () const

Get the enumerated value.

Definition at line 110 of file DemandGenerationMethod.cpp.

Referenced by DemandGenerationMethod().

32.56.4.7 char stdair::DemandGenerationMethod::getMethodAsChar () const

Get the enumerated value as a short string (e.g., 'P' or 'S').

Definition at line 115 of file DemandGenerationMethod.cpp.

32.56.4.8 std::string stdair::DemandGenerationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "P" or "S").

Definition at line 121 of file DemandGenerationMethod.cpp.

32.56.4.9 const std::string stdair::DemandGenerationMethod::describe () const [virtual]

Give a description of the structure (e.g., "PoissonProcess" or "SatisticsOrder").

Implements stdair::StructAbstract.

Definition at line 128 of file DemandGenerationMethod.cpp.

32.56.4.10 bool stdair::DemandGenerationMethod::operator== (const EN_DemandGenerationMethod & *iMethod*) const

Comparison operator.

Definition at line 136 of file DemandGenerationMethod.cpp.

32.56.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.56.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/DemandGenerationMethod.hpp
- stdair/basic/DemandGenerationMethod.cpp

32.57 stdair::DictionaryManager Class Reference

Class wrapper of dictionary business methods.

```
#include <stdair/basic/DictionaryManager.hpp>
```

Static Public Member Functions

- static const stdair::Probability_T keyToValue (const DictionaryKey_T)
- static const DictionaryKey_T valueToKey (const stdair::Probability_T)

32.57.1 Detailed Description

Class wrapper of dictionary business methods.

Definition at line 22 of file DictionaryManager.hpp.

32.57.2 Member Function Documentation

32.57.2.1 const stdair::Probability_T stdair::DictionaryManager::keyToValue (const DictionaryKey_T *iKey*) [static]

Convert from key to value.

Definition at line 12 of file DictionaryManager.cpp.

References stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS.

Referenced by stdair::ContinuousAttributeLite< T >::displayCumulativeDistribution(), stdair::ContinuousAttributeLite< T >::getDerivativeValue(), stdair::ContinuousAttributeLite< T >::getRemainingProportion(), and stdair::ContinuousAttributeLite< T >::getValue().

32.57.2.2 const DictionaryKey_T stdair::DictionaryManager::valueToKey (const stdair::Probability_T *iValue*) [static]

Convert from value to key.

Definition at line 21 of file DictionaryManager.cpp.

References stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS.

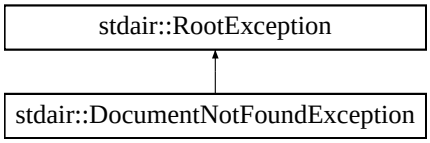
Referenced by stdair::ContinuousAttributeLite< T >::getValue().

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

- stdair/basic/DictionaryManager.hpp
- stdair/basic/DictionaryManager.cpp

32.58 stdair::DocumentNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::DocumentNotFoundException::



Public Member Functions

- DocumentNotFoundException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.58.1 Detailed Description

Document not found.

Definition at line 104 of file stdair_exceptions.hpp.

32.58.2 Constructor & Destructor Documentation

32.58.2.1 stdair::DocumentNotFoundException::DocumentNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 107 of file stdair_exceptions.hpp.

32.58.3 Member Function Documentation

32.58.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.58.4 Member Data Documentation

32.58.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

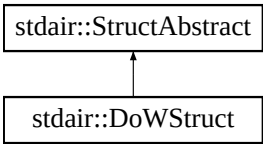
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.59 stdair::DoWStruct Struct Reference

`#include <stdair/bom/DoWStruct.hpp>`Inheritance diagram for stdair::DoWStruct::



Public Types

- typedef std::vector< bool > BooleanList_T

Public Member Functions

- bool getDayOfWeek (const unsigned short i) const
- bool getStandardDayOfWeek (const unsigned short i) const
- void setDayOfWeek (const unsigned short, const bool)
- const std::string describe () const
- const std::string describeShort () const
- DoWStruct shift (const long &) const
- DoWStruct intersection (const DoWStruct &) const
- const bool isValid () const
- DoWStruct (const std::string &iDowString)
- DoWStruct ()
- DoWStruct (const DoWStruct &)
- ~DoWStruct ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

32.59.1 Detailed Description

Define a Day Of the Week (DoW) sequence.

For instance, 1..11.1 means that the period is active on Mon., Thu., Fri. and Sun.

Definition at line 18 of file DoWStruct.hpp.

32.59.2 Member Typedef Documentation

32.59.2.1 typedef std::vector<bool> stdair::DoWStruct::BooleanList_T

Define the bit set representing the DoW.

Definition at line 21 of file DoWStruct.hpp.

32.59.3 Constructor & Destructor Documentation

32.59.3.1 stdair::DoWStruct::DoWStruct (const std::string & *iDowString*)

Constructor from a given bit set (e.g., "0000011" for the week-ends).

Definition at line 21 of file DoWStruct.cpp.

32.59.3.2 stdair::DoWStruct::DoWStruct ()

Default constructors.

Definition at line 14 of file DoWStruct.cpp.

32.59.3.3 stdair::DoWStruct::DoWStruct (const DoWStruct & *iDowStruct*)

Definition at line 34 of file DoWStruct.cpp.

32.59.3.4 stdair::DoWStruct::~DoWStruct () `[inline]`

Default destructor.

Definition at line 63 of file DoWStruct.hpp.

32.59.4 Member Function Documentation

32.59.4.1 bool stdair::DoWStruct::getDayOfWeek (const unsigned short *i*) const

Get the i-th day of the week (Monday being the first one).

Definition at line 66 of file DoWStruct.cpp.

Referenced by intersection(), and isValid().

32.59.4.2 bool stdair::DoWStruct::getStandardDayOfWeek (const unsigned short *i*) const

Get the i-th day of the week (Sunday being the first one).

Definition at line 71 of file DoWStruct.cpp.

32.59.4.3 void stdair::DoWStruct::setDayOfWeek (const unsigned short *i*, const bool *iBool*)

Set the new value for the i-th day-of-week.

Definition at line 82 of file DoWStruct.cpp.

Referenced by intersection(), and shift().

32.59.4.4 const std::string stdair::DoWStruct::describe () const [virtual]

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements stdair::StructAbstract.

Definition at line 52 of file DoWStruct.cpp.

References stdair::DOW_STR.

Referenced by stdair::PeriodStruct::describe().

32.59.4.5 const std::string stdair::DoWStruct::describeShort () const

Display as a bit set (e.g., "1111100").

Definition at line 40 of file DoWStruct.cpp.

Referenced by stdair::PeriodStruct::describeShort().

32.59.4.6 DoWStruct stdair::DoWStruct::shift (const long & *iNbOfDays*) const

Build a new DoW struct by shifting the current DoW by a given number.

Definition at line 88 of file DoWStruct.cpp.

References stdair::DEFAULT_DOW_STRING, and setDayOfWeek().

Referenced by stdair::PeriodStruct::addDateOffset().

32.59.4.7 DoWStruct stdair::DoWStruct::intersection (const DoWStruct & *iDoW*) const

Build a new DoW struct by intersecting two DoW structs.

Definition at line 104 of file DoWStruct.cpp.

References stdair::DEFAULT_DOW_STRING, getDayOfWeek(), and setDayOfWeek().

Referenced by stdair::PeriodStruct::intersection().

32.59.4.8 const bool stdair::DoWStruct::isValid () const

Return if the DoW struct is valid (i.e., has at least one "true").

Definition at line 117 of file DoWStruct.cpp.

References getDayOfWeek().

Referenced by stdair::PeriodStruct::isValid().

32.59.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.59.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

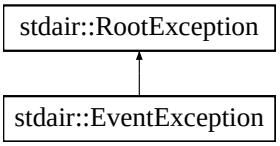
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/bom/DoWStruct.hpp
- stdair/bom/DoWStruct.cpp

32.60 stdair::EventException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::EventException::



Public Member Functions

- EventException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.60.1 Detailed Description

Event.

Definition at line 204 of file stdair_exceptions.hpp.

32.60.2 Constructor & Destructor Documentation

32.60.2.1 stdair::EventException::EventException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 207 of file stdair_exceptions.hpp.

32.60.3 Member Function Documentation

32.60.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.60.4 Member Data Documentation

32.60.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

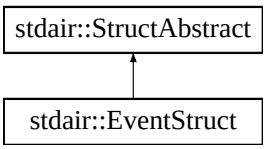
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.61 stdair::EventStruct Struct Reference

`#include <stdair/bom/EventStruct.hpp>`Inheritance diagram for stdair::EventStruct::



Public Member Functions

- const EventType::EN_EventType & getEventType () const
- const LongDuration_T & getEventTimeStamp () const
- const DateTime_T & getEventTime () const
- const BookingRequestStruct & getBookingRequest () const
- const CancellationStruct & getCancellation () const
- const OptimisationNotificationStruct & getOptimisationNotificationStruct () const
- const SnapshotStruct & getSnapshotStruct () const
- const RMEventStruct & getRMEvent () const
- const BreakPointStruct & getBreakPoint () const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- EventStruct ()
- EventStruct (const EventType::EN_EventType &, BookingRequestPtr_T)
- EventStruct (const EventType::EN_EventType &, CancellationPtr_T)
- EventStruct (const EventType::EN_EventType &, const DateTime_T &iDCPDate, OptimisationNotificationPtr_T)
- EventStruct (const EventType::EN_EventType &, SnapshotPtr_T)
- EventStruct (const EventType::EN_EventType &, RMEventPtr_T)
- EventStruct (const EventType::EN_EventType &, BreakPointPtr_T)
- EventStruct (const EventStruct &)
- ~EventStruct ()
- void incrementEventTimeStamp ()
- void toStream (std::ostream &ioOut) const

32.61.1 Detailed Description

Structure holding the details of an event.

Note:

No event should be scheduled before the date-time corresponding to the DEFAULT_EVENT_-OLDEST_DATETIME constant (as of Feb. 2011, that date is set to Jan. 1, 2010). That constant is specified in the stdair/basic/BasConst.cpp file. In other words, the simulation should not specified to start before that date-time.

Definition at line 36 of file EventStruct.hpp.

32.61.2 Constructor & Destructor Documentation

32.61.2.1 stdair::EventStruct::EventStruct ()

Default constructor.

Definition at line 26 of file EventStruct.cpp.

32.61.2.2 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, BookingRequestPtr_T *ioRequestPtr*)

Constructor for events corresponding to booking requests.

Definition at line 31 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.3 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, CancellationPtr_T *ioCancellationPtr*)

Constructor for events corresponding to cancellations.

Definition at line 55 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.4 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, const DateTime_T & *iDCPDate*, OptimisationNotificationPtr_T *ioOptimisationNotificationPtr*)

Constructor for events corresponding to optimisation requests.

Definition at line 80 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.5 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, SnapshotPtr_T *ioSnapshotPtr*)

Constructor for events corresponding to snapshot requests.

Definition at line 105 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.6 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, RMEventPtr_T *ioRMEventPtr*)

Constructor for events corresponding to RM events.

Definition at line 130 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.7 stdair::EventStruct::EventStruct (const EventType::EN_EventType & *iEventType*, BreakPointPtr_T *ioBreakPointPtr*)

Constructor for events corresponding to Break Point events.

Definition at line 155 of file EventStruct.cpp.

References stdair::DEFAULT_EVENT_OLDEST_DATETIME.

32.61.2.8 stdair::EventStruct::EventStruct (const EventStruct & *iEventStruct*)

Copy constructor.

Definition at line 180 of file EventStruct.cpp.

32.61.2.9 stdair::EventStruct::~EventStruct ()

Destructor.

Definition at line 243 of file EventStruct.cpp.

32.61.3 Member Function Documentation

32.61.3.1 const EventType::EN_EventType& stdair::EventStruct::getEventType () const [inline]

Get the event type

Definition at line 41 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject(), stdair::BomJSONExport::jsonExportBreakPointObject(), and stdair::STDAIR_Service::jsonExportEventObject().

32.61.3.2 const LongDuration_T& stdair::EventStruct::getEventTimeStamp () const [inline]

Get the event time stamp

Definition at line 46 of file EventStruct.hpp.

32.61.3.3 const DateTime_T & stdair::EventStruct::getEventTime () const

Get the event date-time i.e the stamp converted to a date-time format.

Definition at line 311 of file EventStruct.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::DEFAULT_EVENT_OLDEST_DATETIME, stdair::EventType::OPT_NOT_4_FD, stdair::EventType::RM, and stdair::EventType::SNAPSHOT.

32.61.3.4 const BookingRequestStruct& stdair::EventStruct::getBookingRequest () const [inline]

Get a reference on the booking request referred to by event.

Note:

When that event is not of type booking request (EventType::BKG_REQ), an assertion fails.

Definition at line 59 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBookingRequestObject().

32.61.3.5 const CancellationStruct& stdair::EventStruct::getCancellation () const [inline]

Get a reference on the cancellation referred to by event.

Note:

When that event is not of type cancellation (EventType::CX), an assertion fails.

Definition at line 70 of file EventStruct.hpp.

32.61.3.6 const OptimisationNotificationStruct& stdair::EventStruct::getOptimisationNotificationStruct () const [inline]

Get a reference on the optimisation notification referred to by event.

Note:

When that event is not of type optimisation notification for optimisation notification (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 83 of file EventStruct.hpp.

32.61.3.7 const SnapshotStruct& stdair::EventStruct::getSnapshotStruct () const [inline]

Get a reference on the snapshot referred to by event.

Note:

When that event is not of type snapshot for snapshot (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 95 of file EventStruct.hpp.

32.61.3.8 const RMEventStruct& stdair::EventStruct::getRMEvent () const [inline]

Get a reference on the RM event referred to by the generic event.

Note:

When that event is not of type RM event for snapshot (EventType::OPT_NOT_4_FD), an assertion fails.

Definition at line 107 of file EventStruct.hpp.

32.61.3.9 const BreakPointStruct& stdair::EventStruct::getBreakPoint () const [inline]

Get a reference on the break point referred to by event.

Note:

When that event is not of type booking break point (EventType::BRK_PT), an assertion fails.

Definition at line 118 of file EventStruct.hpp.

Referenced by stdair::BomJSONExport::jsonExportBreakPointObject().

32.61.3.10 void stdair::EventStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 247 of file EventStruct.cpp.

32.61.3.11 const std::string stdair::EventStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 251 of file EventStruct.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::DEFAULT_EVENT_OLDEST_DATETIME, stdair::EventType::getLabel(), stdair::EventType::OPT_NOT_4_FD, stdair::EventType::RM, and stdair::EventType::SNAPSHOT.

32.61.3.12 void stdair::EventStruct::incrementEventTimeStamp ()

Increment the date-time stamp which is counted in milliseconds.

This incrementation of one millisecond is needed when the insertion in the event queue failed, that is to say when an event with the exact same time stamp has already been inserted in the queue.

Definition at line 357 of file EventStruct.cpp.

32.61.3.13 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

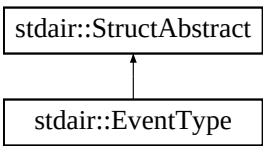
References stdair::StructAbstract::describe().

The documentation for this struct was generated from the following files:

- stdair/bom/EventStruct.hpp
- stdair/bom/EventStruct.cpp

32.62 stdair::EventType Struct Reference

`#include <stdair/basic/EventType.hpp>`Inheritance diagram for stdair::EventType::



Public Types

- enum EN_EventType {
  BKG_REQ = 0, CX, OPT_NOT_4_FD, OPT_NOT_4_NET,
  SKD_CHG, SNAPSHOT, RM, BRK_PT,
  LAST_VALUE }

Public Member Functions

- EN_EventType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_EventType &) const
- EventType (const EN_EventType &)
- EventType (const char iType)
- EventType (const std::string &iTypeStr)
- EventType (const EventType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_EventType &)
- static char getTypeLabel (const EN_EventType &)
- static std::string getTypeLabelAsString (const EN_EventType &)
- static std::string describeLabels ()

32.62.1 Detailed Description

Enumeration of event types.

Definition at line 15 of file EventType.hpp.

32.62.2 Member Enumeration Documentation

32.62.2.1 enum stdair::EventType::EN_EventType

Enumerator:

BKG_REQ
CX
OPT_NOT_4_FD
OPT_NOT_4_NET
SKD_CHG
SNAPSHOT
RM
BRK_PT
LAST_VALUE

Definition at line 17 of file EventType.hpp.

32.62.3 Constructor & Destructor Documentation

32.62.3.1 stdair::EventType::EventType (const EN_EventType & *iEventType*)

Constructor.

Definition at line 36 of file EventType.cpp.

32.62.3.2 stdair::EventType::EventType (const char *iType*)

Constructor using a char.

Definition at line 41 of file EventType.cpp.

References BKG_REQ, BRK_PT, CX, describeLabels(), LAST_VALUE, OPT_NOT_4_FD, OPT_NOT_-4_NET, RM, SKD_CHG, and SNAPSHOT.

32.62.3.3 stdair::EventType::EventType (const std::string & *iTypeStr*)

Constructor using a string.

Definition at line 64 of file EventType.cpp.

References describeLabels(), and LAST_VALUE.

32.62.3.4 stdair::EventType::EventType (const EventType & *iEventType*)

Default copy constructor.

Definition at line 31 of file EventType.cpp.

32.62.4 Member Function Documentation

32.62.4.1 const std::string & stdair::EventType::getLabel (const EN_EventType & *iType*) `[static]`

Get the label as a string (e.g., "BookingRe-quest", "Cancellation", "OptimisationNotificationForFlightDate", "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot", "RevenueManagement", "BreakPoint" or "BookingRequest").

Definition at line 83 of file EventType.cpp.

Referenced by stdair::EventStruct::describe(), stdair::BomJSONExport::jsonExportBookingRequestObject(), and stdair::BomJSONExport::jsonExportBreakPointObject().

32.62.4.2 char stdair::EventType::getTypeLabel (const EN_EventType & *iType*) [static]

Get the label as a single char (e.g., 'B', 'X', 'F', 'N', 'C', 'S', 'R' or 'P').

Definition at line 88 of file EventType.cpp.

32.62.4.3 std::string stdair::EventType::getTypeLabelAsString (const EN_EventType & *iType*) [static]

Get the label as a string of a single char (e.g., "B", "X", "F", "N", "C", "S", "R" or "P").

Definition at line 93 of file EventType.cpp.

32.62.4.4 std::string stdair::EventType::describeLabels () [static]

List the labels.

Definition at line 100 of file EventType.cpp.

References LAST_VALUE.

Referenced by EventType().

32.62.4.5 EventType::EN_EventType stdair::EventType::getType () const

Get the enumerated value.

Definition at line 112 of file EventType.cpp.

32.62.4.6 std::string stdair::EventType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "B", "X", "F", "N", "C", "S", "R" or "P").

Definition at line 117 of file EventType.cpp.

32.62.4.7 const std::string stdair::EventType::describe () const [virtual]

Give a description of the structure (e.g., "BookingRequest", "Cancellation", "OptimisationNotificationForFlightDate", "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot", "RevenueManagement", "BreakPoint" or "BookingRequest").

Implements stdair::StructAbstract.

Definition at line 124 of file EventType.cpp.

32.62.4.8 bool stdair::EventType::operator== (const EN_EventType & *iType*) const

Comparison operator.

Definition at line 131 of file EventType.cpp.

32.62.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.62.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

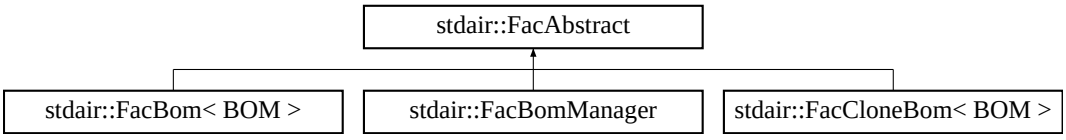
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/EventType.hpp
- stdair/basic/EventType.cpp

32.63 stdair::FacAbstract Class Reference

`#include <stdair/factory/FacAbstract.hpp>`Inheritance diagram for stdair::FacAbstract::



Public Member Functions

- virtual ~FacAbstract ()

Protected Member Functions

- FacAbstract ()

32.63.1 Detailed Description

Base class for Factory layer.

Definition at line 10 of file FacAbstract.hpp.

32.63.2 Constructor & Destructor Documentation

32.63.2.1 stdair::FacAbstract::~FacAbstract () `[virtual]`

Destructor.

Definition at line 13 of file FacAbstract.cpp.

32.63.2.2 stdair::FacAbstract::FacAbstract () `[inline, protected]`

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 18 of file FacAbstract.hpp.

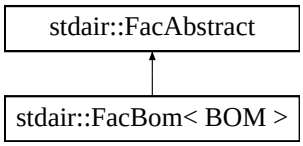
The documentation for this class was generated from the following files:

- stdair/factory/FacAbstract.hpp
- stdair/factory/FacAbstract.cpp

32.64 stdair::FacBom< BOM > Class Template Reference

Base class for Factory layer.

`#include <stdair/factory/FacBom.hpp>`Inheritance diagram for stdair::FacBom< BOM >::



Public Member Functions

- BOM & create ()
- BOM & create (const Key_T &)
- BOM & create (const BOM &)

- ~FacBom ()
- void clean ()

Static Public Member Functions

- static FacBom & instance ()

Protected Member Functions

- FacBom ()

32.64.1 Detailed Description

template<typename BOM> class stdair::FacBom< BOM >

Base class for Factory layer.

Definition at line 22 of file FacBom.hpp.

32.64.2 Constructor & Destructor Documentation

32.64.2.1 template<typename BOM> stdair::FacBom< BOM >::FacBom () [inline, protected]

Default Constructor.

Definition at line 50 of file FacBom.hpp.

Referenced by stdair::FacBom< BOM >::instance().

32.64.2.2 template<typename BOM> stdair::FacBom< BOM >::~FacBom () [inline]

Destructor.

Definition at line 56 of file FacBom.hpp.

References stdair::FacBom< BOM >::clean().

32.64.3 Member Function Documentation

32.64.3.1 template<typename BOM > FacBom< BOM > & stdair::FacBom< BOM >::instance () [inline, static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacBom&

Definition at line 84 of file FacBom.hpp.

References stdair::FacBom< BOM >::FacBom().

Referenced by stdair::FacBom< BOM >::create().

32.64.3.2 template<typename BOM > BOM & stdair::FacBom< BOM >::create () [inline]

Create a BOM object, given a key or not.

Definition at line 112 of file FacBom.hpp.

References stdair::FacBom< BOM >::create(), and stdair::FacBom< BOM >::instance().

Referenced by stdair::FacBom< BOM >::create().

32.64.3.3 template<typename BOM > BOM & stdair::FacBom< BOM >::create (const Key_T & *iKey*) [inline]

Definition at line 118 of file FacBom.hpp.

32.64.3.4 template<typename BOM > BOM & stdair::FacBom< BOM >::create (const BOM & *iBom*) [inline]

Definition at line 126 of file FacBom.hpp.

32.64.3.5 template<typename BOM > void stdair::FacBom< BOM >::clean () [inline]

Destroyed all the object instantiated by this factory.

Definition at line 95 of file FacBom.hpp.

Referenced by stdair::FacBom< BOM >::~FacBom().

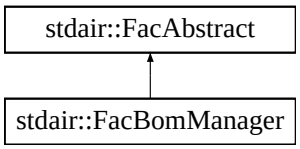
The documentation for this class was generated from the following file:

- stdair/factory/FacBom.hpp

32.65 stdair::FacBomManager Class Reference

Utility class for linking StdAir-based objects.

`#include <stdair/factory/FacBomManager.hpp>`Inheritance diagram for stdair::FacBomManager::



Public Member Functions

- ~FacBomManager ()

- template<>
  void addToList (SegmentDate &ioSegmentDate, SegmentDate &ioMarketingSegmentDate)

Static Public Member Functions

- template<typename OBJECT2 , typename OBJECT1 >
  static BomHolder< OBJECT2 > * getBomHolderPtr (OBJECT1 &)
- template<typename OBJECT2 , typename OBJECT1 >
  static BomHolder< OBJECT2 > & addBomHolder (OBJECT1 &)
- template<typename OBJECT1 , typename OBJECT2 >
  static void addToList (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1 , typename OBJECT2 >
  static void addToMap (OBJECT1 &, OBJECT2 &, const MapKey_T &)
- template<typename OBJECT1 , typename OBJECT2 >
  static void addToMap (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1 , typename OBJECT2 >
  static void addToListAndMap (OBJECT1 &, OBJECT2 &)
- template<typename OBJECT1 , typename OBJECT2 >
  static void addToListAndMap (OBJECT1 &, OBJECT2 &, const MapKey_T &)
- template<typename PARENT , typename CHILD >
  static void linkWithParent (PARENT &, CHILD &)
- template<typename OBJECT2 , typename OBJECT1 >
  static void cloneHolder (OBJECT1 &, const OBJECT1 &)
- static void resetYieldBasedNestingStructure (const SegmentCabin &)
- static void setAirlineFeature (Inventory &iInventory, AirlineFeature &iAirlineFeature)
- static void linkWithOperating (SegmentDate &iSegmentDate, SegmentDate &iOperatingSegmentDate)

Protected Member Functions

- FacBomManager ()

32.65.1 Detailed Description

Utility class for linking StdAir-based objects.

Definition at line 30 of file FacBomManager.hpp.

32.65.2 Constructor & Destructor Documentation

32.65.2.1 stdair::FacBomManager::FacBomManager () [inline, protected]

Default Constructor.

This constructor is protected to comply with the singleton pattern.

Definition at line 225 of file FacBomManager.hpp.

32.65.2.2 stdair::FacBomManager::~FacBomManager () [inline]

Destructor.

Definition at line 231 of file FacBomManager.hpp.

32.65.3 Member Function Documentation

32.65.3.1 template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 > * stdair::FacBomManager::getBomHolderPtr (OBJECT1 & *ioObject1*) [inline, static]

Retrieve a pointer on the holder of children (OBJECT2 type) for the parent (OBJECT1 type). If the holder does not exist, return NULL.

Parameters:

typename OBJECT1& Parent object.

Returns:

typename BomHolder<OBJECT2>* BomHolder for the children objects.

Definition at line 268 of file FacBomManager.hpp.

32.65.3.2 template<typename OBJECT2 , typename OBJECT1 > BomHolder< OBJECT2 > & stdair::FacBomManager::addBomHolder (OBJECT1 & *ioObject1*) [inline, static]

Instantiate a BomHolder<OBJECT2> object, add it to the OBJECT1-typed object, given as parameter, and return a reference on that newly created BomHolder.

Parameters:

typename OBJECT1& Parent object.

Returns:

typename BomHolder<OBJECT2>& Just created BomHolder (e.g., for the children objects).

Definition at line 238 of file FacBomManager.hpp.

32.65.3.3 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToList (OBJECT1 & *ioObject1*, OBJECT2 & *ioObject2*) [inline, static]

Add an OBJECT2-typed object (typically, a child) to the dedicated list held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying list is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 354 of file FacBomManager.hpp.

32.65.3.4 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToMap (OBJECT1 & *ioObject1*, OBJECT2 & *ioObject2*, const MapKey_T & *iKey*) `[inline, static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

const MapKey_T&

Definition at line 424 of file FacBomManager.hpp.

Referenced by addToMap().

32.65.3.5 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToMap (OBJECT1 & *ioObject1*, OBJECT2 & *ioObject2*) `[inline, static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated map held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying map is actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 446 of file FacBomManager.hpp.

References addToMap().

32.65.3.6 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToListAndMap (OBJECT1 & *ioObject1*, OBJECT2 & *ioObject2*) `[inline, static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 490 of file FacBomManager.hpp.

32.65.3.7 template<typename OBJECT1 , typename OBJECT2 > void stdair::FacBomManager::addToListAndMap (OBJECT1 & *ioObject1*, OBJECT2 & *ioObject2*, const MapKey_T & *iKey*) `[inline, static]`

Add an OBJECT2-typed object (typically, a child) to the dedicated containers (list and map) held by the OBJECT1-typed object (typically, a parent).

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

const MapKey_T&

Definition at line 467 of file FacBomManager.hpp.

32.65.3.8 template<typename PARENT , typename CHILD > void stdair::FacBomManager::linkWithParent (PARENT & *ioParent*, CHILD & *ioChild*) `[inline, static]`

Allow the CHILD object to store a pointer on its PARENT object.

Parameters:

typename PARENT& Parent object.

typename CHILD& Child object.

Definition at line 511 of file FacBomManager.hpp.

Referenced by stdair::serialiseHelper().

32.65.3.9 template<typename OBJECT2 , typename OBJECT1 > void stdair::FacBomManager::cloneHolder (OBJECT1 & *ioDest*, const OBJECT1 & *iOri*) `[inline, static]`

Clone the underlying containers (held by the BomHolder<OBJECT2>-typed holder) of the OBJECT1-typed object.

Note:

The underlying containers are actually stored within an object of type BomHolder<OBJECT2>.

Parameters:

typename OBJECT1& Parent object.

typename OBJECT2& Child object.

Definition at line 519 of file FacBomManager.hpp.

References stdair::BomHolder< BOM >::_bomList, and stdair::BomHolder< BOM >::_bomMap.

32.65.3.10 void stdair::FacBomManager::resetYieldBasedNestingStructure (const SegmentCabin & *iSegmentCabin*) [static]

Reset the yield-based nesting structure of a segment-cabin. This method is used with FA or MRT.

Definition at line 20 of file FacBomManager.cpp.

References stdair::BomHolder< BOM >::_bomList, stdair::NestingNode::describeKey(), stdair::NestingNode::getHolderMap(), stdair::NestingNode::setYield(), and stdair::YIELD_BASED_-NESTING_STRUCTURE_CODE.

32.65.3.11 static void stdair::FacBomManager::setAirlineFeature (Inventory & *iInventory*, AirlineFeature & *iAirlineFeature*) [inline, static]

Set the airline feature object of an inventory.

Definition at line 205 of file FacBomManager.hpp.

32.65.3.12 static void stdair::FacBomManager::linkWithOperating (SegmentDate & *iSegmentDate*, SegmentDate & *iOperatingSegmentDate*) [inline, static]

Link the segment date with its operating segment date.

Definition at line 213 of file FacBomManager.hpp.

32.65.3.13 template<> void stdair::FacBomManager::addToList (SegmentDate & *ioSegmentDate*, SegmentDate & *ioMarketingSegmentDate*) [inline]

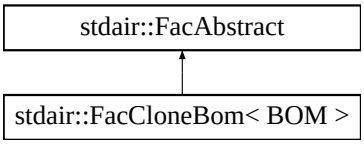
The documentation for this class was generated from the following files:

- stdair/factory/FacBomManager.hpp
- stdair/factory/FacBomManager.cpp

32.66 stdair::FacCloneBom< BOM > Class Template Reference

Base class for Factory layer.

`#include <stdair/factory/FacCloneBom.hpp>`Inheritance diagram for stdair::FacCloneBom< BOM >::



Public Member Functions

- BOM & clone (const BOM &)
- ~FacCloneBom ()
- void clean ()

Static Public Member Functions

- static FacCloneBom & instance ()

Protected Member Functions

- FacCloneBom ()

32.66.1 Detailed Description

template<typename BOM> class stdair::FacCloneBom< BOM >

Base class for Factory layer.

Definition at line 22 of file FacCloneBom.hpp.

32.66.2 Constructor & Destructor Documentation

32.66.2.1 template<typename BOM > stdair::FacCloneBom< BOM >::FacCloneBom () [inline, protected]

Default Constructor.

Definition at line 48 of file FacCloneBom.hpp.

Referenced by stdair::FacCloneBom< BOM >::instance().

32.66.2.2 template<typename BOM > stdair::FacCloneBom< BOM >::~FacCloneBom () [inline]

Destructor.

Definition at line 54 of file FacCloneBom.hpp.

References stdair::FacCloneBom< BOM >::clean().

32.66.3 Member Function Documentation

32.66.3.1 template<typename BOM > FacCloneBom< BOM > & stdair::FacCloneBom< BOM >::instance () [inline, static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacCloneBom&

Definition at line 82 of file FacCloneBom.hpp.

References stdair::FacCloneBom< BOM >::FacCloneBom().

32.66.3.2 template<typename BOM > BOM & stdair::FacCloneBom< BOM >::clone (const BOM & *iBom*) [inline]

Clone a BOM object.

Definition at line 110 of file FacCloneBom.hpp.

32.66.3.3 template<typename BOM > void stdair::FacCloneBom< BOM >::clean () [inline]

Destroyed all the object instantiated by this factory.

Definition at line 93 of file FacCloneBom.hpp.

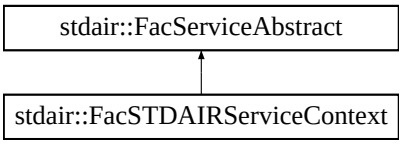
Referenced by stdair::FacCloneBom< BOM >::~FacCloneBom().

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

- stdair/factory/FacCloneBom.hpp

32.67 stdair::FacServiceAbstract Class Reference

`#include <stdair/service/FacServiceAbstract.hpp>`Inheritance diagram for stdair::FacServiceAbstract::



Public Types

- typedef std::vector< ServiceAbstract * > ServicePool_T

Public Member Functions

- virtual ~FacServiceAbstract ()
- void clean ()

Protected Member Functions

- FacServiceAbstract ()

Protected Attributes

- ServicePool_T _pool

32.67.1 Detailed Description

Base class for the (Service) Factory layer.

Definition at line 16 of file FacServiceAbstract.hpp.

32.67.2 Member Typedef Documentation

32.67.2.1 typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

32.67.3 Constructor & Destructor Documentation

32.67.3.1 stdair::FacServiceAbstract::~FacServiceAbstract () [virtual]

Destructor.

Definition at line 13 of file FacServiceAbstract.cpp.

References clean().

32.67.3.2 stdair::FacServiceAbstract::FacServiceAbstract () [inline, protected]

Default Constructor.

This constructor is protected to ensure the class is abstract.

Definition at line 31 of file FacServiceAbstract.hpp.

32.67.4 Member Function Documentation

32.67.4.1 void stdair::FacServiceAbstract::clean ()

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References _pool.

Referenced by ~FacServiceAbstract().

32.67.5 Member Data Documentation

32.67.5.1 ServicePool_T stdair::FacServiceAbstract::_pool [protected]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by clean(), and stdair::FacSTDAIRServiceContext::create().

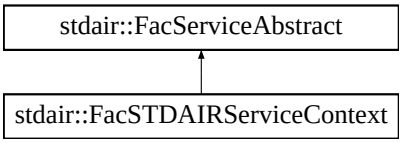
The documentation for this class was generated from the following files:

- stdair/service/FacServiceAbstract.hpp
- stdair/service/FacServiceAbstract.cpp

32.68 stdair::FacSTDAIRServiceContext Class Reference

Factory for Bucket.

`#include <stdair/service/FacSTDAIRServiceContext.hpp>`Inheritance diagram for stdair::FacSTDAIRServiceContext::



Public Types

- typedef std::vector< ServiceAbstract * > ServicePool_T

Public Member Functions

- ~FacSTDAIRServiceContext ()
- STDAIR_ServiceContext & create ()
- void clean ()

Static Public Member Functions

- static FacSTDAIRServiceContext & instance ()

Protected Member Functions

- FacSTDAIRServiceContext ()

Protected Attributes

- ServicePool_T _pool

32.68.1 Detailed Description

Factory for Bucket.

Definition at line 18 of file FacSTDAIRServiceContext.hpp.

32.68.2 Member Typedef Documentation

32.68.2.1 typedef std::vector<ServiceAbstract*> stdair::FacServiceAbstract::ServicePool_T [inherited]

Define the list (pool) of Service objects.

Definition at line 20 of file FacServiceAbstract.hpp.

32.68.3 Constructor & Destructor Documentation

32.68.3.1 stdair::FacSTDAIRServiceContext::~FacSTDAIRServiceContext ()

Destructor.

The Destruction put the _instance to NULL in order to be clean for the next FacSTDAIRServiceContext::instance().

Definition at line 16 of file FacSTDAIRServiceContext.cpp.

32.68.3.2 stdair::FacSTDAIRServiceContext::FacSTDAIRServiceContext () [inline, protected]

Default Constructor.

This constructor is protected in order to ensure the singleton pattern.

Definition at line 54 of file FacSTDAIRServiceContext.hpp.

Referenced by instance().

32.68.4 Member Function Documentation

32.68.4.1 FacSTDAIRServiceContext & stdair::FacSTDAIRServiceContext::instance () [static]

Provide the unique instance.

The singleton is instantiated when first used.

Returns:

FacSTDAIRServiceContext&

Definition at line 21 of file FacSTDAIRServiceContext.cpp.

References FacSTDAIRServiceContext().

32.68.4.2 STDAIR_ServiceContext & stdair::FacSTDAIRServiceContext::create ()

Create a new STDAIR_ServiceContext object.

This new object is added to the list of instantiated objects.

Returns:

STDAIR_ServiceContext& The newly created object.

Definition at line 33 of file FacSTDAIRServiceContext.cpp.

References stdair::FacServiceAbstract::_pool.

32.68.4.3 void stdair::FacServiceAbstract::clean () [inherited]

Destroyed all the object instantiated by this factory.

Definition at line 18 of file FacServiceAbstract.cpp.

References stdair::FacServiceAbstract::_pool.

Referenced by stdair::FacServiceAbstract::~FacServiceAbstract().

32.68.5 Member Data Documentation

32.68.5.1 ServicePool_T stdair::FacServiceAbstract::_pool [protected, inherited]

List of instantiated Business Objects

Definition at line 34 of file FacServiceAbstract.hpp.

Referenced by stdair::FacServiceAbstract::clean(), and create().

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

- stdair/service/FacSTDAIRServiceContext.hpp
- stdair/service/FacSTDAIRServiceContext.cpp

32.69 stdair::FacSupervisor Class Reference

```
#include <stdair/service/FacSupervisor.hpp>
```

Public Types

- typedef std::list< FacAbstract * > PersistentBomFactoryPool_T
- typedef std::list< FacAbstract * > CloneBomFactoryPool_T
- typedef std::list< FacServiceAbstract * > ServiceFactoryPool_T

Public Member Functions

- void registerPersistentBomFactory (FacAbstract *)
- void registerCloneBomFactory (FacAbstract *)
- void registerServiceFactory (FacServiceAbstract *)
- void cleanPersistentBomLayer ()
- void cleanCloneBomLayer ()
- void cleanServiceLayer ()
- ~FacSupervisor ()

Static Public Member Functions

- static FacSupervisor & instance ()
- static void cleanLoggerService ()
- static void cleanDBSessionManager ()
- static void cleanAll ()

Protected Member Functions

- FacSupervisor ()
- FacSupervisor (const FacSupervisor &)

32.69.1 Detailed Description

Singleton class to register and clean all Factories.

Definition at line 20 of file FacSupervisor.hpp.

32.69.2 Member Typedef Documentation

32.69.2.1 typedef std::list<FacAbstract*> stdair::FacSupervisor::PersistentBomFactoryPool_T

Define the pool (list) of factories.

Definition at line 25 of file FacSupervisor.hpp.

32.69.2.2 typedef std::list<FacAbstract*> stdair::FacSupervisor::CloneBomFactoryPool_T

Definition at line 26 of file FacSupervisor.hpp.

32.69.2.3 typedef std::list<FacServiceAbstract*> stdair::FacSupervisor::ServiceFactoryPool_T

Definition at line 27 of file FacSupervisor.hpp.

32.69.3 Constructor & Destructor Documentation

32.69.3.1 stdair::FacSupervisor::~FacSupervisor ()

Destructor.

That destructors is applied on the static instance. It then deletes in turn all the other registered objects.

Definition at line 27 of file FacSupervisor.cpp.

References cleanCloneBomLayer(), cleanPersistentBomLayer(), and cleanServiceLayer().

32.69.3.2 stdair::FacSupervisor::FacSupervisor () [inline, protected]

Default Constructor.

This constructor is protected to ensure the singleton pattern.

Definition at line 120 of file FacSupervisor.hpp.

Referenced by instance().

32.69.3.3 stdair::FacSupervisor::FacSupervisor (const FacSupervisor &) [inline, protected]

Definition at line 121 of file FacSupervisor.hpp.

32.69.4 Member Function Documentation

32.69.4.1 FacSupervisor & stdair::FacSupervisor::instance () [static]

Provide the unique (static) instance of the FacSupervisor object.

The singleton is instantiated when first used.

Returns:

FacSupervisor&

Definition at line 18 of file FacSupervisor.cpp.

References FacSupervisor().

Referenced by stdair::STDAIR_Service::clonePersistentBom().

32.69.4.2 void stdair::FacSupervisor::registerPersistentBomFactory (FacAbstract * *ioFac_ptr*)

Register a newly instantiated persistent factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor

Parameters:

FacAbstract* The concrete Factory to register.

Definition at line 34 of file FacSupervisor.cpp.

32.69.4.3 void stdair::FacSupervisor::registerCloneBomFactory (FacAbstract * *ioFac_ptr*)

Register a newly instantiated concrete factory for the Bom layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor

Parameters:

FacAbstract* The concrete Factory to register.

Definition at line 39 of file FacSupervisor.cpp.

32.69.4.4 void stdair::FacSupervisor::registerServiceFactory (FacServiceAbstract * *ioFac_ptr*)

Register a newly instantiated concrete factory for the Service layer.

When a concrete Factory is firstly instantiated this factory have to register itself to the FacSupervisor.

Parameters:

FacServiceAbstract& the concrete Factory to register.

Definition at line 44 of file FacSupervisor.cpp.

32.69.4.5 void stdair::FacSupervisor::cleanPersistentBomLayer ()

Clean all the persistent registered object.

Call the clean method of all the instantiated persistent factories for the BomStructure layer.

Definition at line 49 of file FacSupervisor.cpp.

Referenced by ~FacSupervisor().

32.69.4.6 void stdair::FacSupervisor::cleanCloneBomLayer ()

Clean all the clone registered object.

Call the clean method of all the instantiated factories for the BomStructure layer.

Definition at line 62 of file FacSupervisor.cpp.

Referenced by stdair::STDAIR_Service::clonePersistentBom(), and ~FacSupervisor().

32.69.4.7 void stdair::FacSupervisor::cleanServiceLayer ()

Clean all Service created object.

Call the clean method of all the instantiated factories for the Service layer.

Definition at line 76 of file FacSupervisor.cpp.

Referenced by ~FacSupervisor().

32.69.4.8 void stdair::FacSupervisor::cleanLoggerService () `[static]`

Delete the static instance of the Logger object.

Definition at line 90 of file FacSupervisor.cpp.

Referenced by cleanAll().

32.69.4.9 void stdair::FacSupervisor::cleanDBSessionManager () `[static]`

Delete the static instance of the DBSessionManager object.

Definition at line 96 of file FacSupervisor.cpp.

Referenced by cleanAll().

32.69.4.10 void stdair::FacSupervisor::cleanAll () [static]

Clean the static instance. As the static instance (singleton) is deleted, all the other registered objects will be deleted in turn.

Definition at line 102 of file FacSupervisor.cpp.

References cleanDBSessionManager(), and cleanLoggerService().

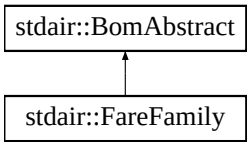
The documentation for this class was generated from the following files:

- stdair/service/FacSupervisor.hpp
- stdair/service/FacSupervisor.cpp

32.70 stdair::FareFamily Class Reference

Class representing the actual attributes for a family fare.

`#include <stdair/bom/FareFamily.hpp>`Inheritance diagram for stdair::FareFamily::



Public Types

- typedef FareFamilyKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FamilyCode_T & getFamilyCode () const
- const HolderMap_T & getHolderMap () const
- const FRAT5Curve_T & getFrat5Curve () const
- const FFDisutilityCurve_T & getDisutilityCurve () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const MeanStdDevPairVector_T & getMeanStdDev () const
- void setFrat5Curve (const FRAT5Curve_T &iFRAT5Curve)
- void setDisutilityCurve (const FFDisutilityCurve_T &iDisutilityCurve)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setMeanStdDev (const MeanStdDevPairVector_T &iMeanStdDev)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Public Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- FRAT5Curve_T _frat5Curve
- FFDisutilityCurve_T _disutilityCurve
- MeanValue_T _mean
- StdDevValue_T _stdDev
- MeanStdDevPairVector_T _meanStdDev

Protected Member Functions

- FareFamily (const Key_T &)
- virtual ~FareFamily ()

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.70.1 Detailed Description

Class representing the actual attributes for a family fare.

Definition at line 28 of file FareFamily.hpp.

32.70.2 Member Typedef Documentation

32.70.2.1 typedef FareFamilyKey stdair::FareFamily::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file FareFamily.hpp.

32.70.3 Constructor & Destructor Documentation

32.70.3.1 stdair::FareFamily::FareFamily (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 32 of file FareFamily.cpp.

32.70.3.2 stdair::FareFamily::~FareFamily () `[protected, virtual]`

Destructor.

Definition at line 36 of file FareFamily.cpp.

32.70.4 Member Function Documentation

32.70.4.1 const Key_T& stdair::FareFamily::getKey () const [inline]

Get the family fare key.

Definition at line 45 of file FareFamily.hpp.

References _key.

32.70.4.2 BomAbstract* const stdair::FareFamily::getParent () const [inline]

Get the parent object.

Definition at line 50 of file FareFamily.hpp.

References _parent.

32.70.4.3 const FamilyCode_T& stdair::FareFamily::getFamilyCode () const [inline]

Get the family fare code (part of the primary key).

Definition at line 55 of file FareFamily.hpp.

References _key, and stdair::FareFamilyKey::getFamilyCode().

32.70.4.4 const HolderMap_T& stdair::FareFamily::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 60 of file FareFamily.hpp.

References _holderMap.

32.70.4.5 const FRAT5Curve_T& stdair::FareFamily::getFrat5Curve () const [inline]

Get the FRAT5 Curve.

Definition at line 65 of file FareFamily.hpp.

References _frat5Curve.

32.70.4.6 const FFDisutilityCurve_T& stdair::FareFamily::getDisutilityCurve () const [inline]

Get the Disutility Curve.

Definition at line 70 of file FareFamily.hpp.

References _disutilityCurve.

32.70.4.7 const MeanValue_T& stdair::FareFamily::getMean () const [inline]

Demand distribution.

Definition at line 75 of file FareFamily.hpp.

References _mean.

32.70.4.8 const StdDevValue_T& stdair::FareFamily::getStdDev () const [inline]

Definition at line 76 of file FareFamily.hpp.

References _stdDev.

32.70.4.9 const MeanStdDevPairVector_T& stdair::FareFamily::getMeanStdDev () const [inline]

Demand distribution.

Definition at line 79 of file FareFamily.hpp.

References _meanStdDev.

32.70.4.10 void stdair::FareFamily::setFrat5Curve (const FRAT5Curve_T & *iFRAT5Curve*) [inline]

FRAT5 Curve.

Definition at line 85 of file FareFamily.hpp.

References _frat5Curve.

32.70.4.11 void stdair::FareFamily::setDisutilityCurve (const FFDisutilityCurve_T & *iDisutilityCurve*) [inline]

Disutility Curve.

Definition at line 90 of file FareFamily.hpp.

References _disutilityCurve.

32.70.4.12 void stdair::FareFamily::setMean (const MeanValue_T & *iMean*) [inline]

Demand distribution.

Definition at line 95 of file FareFamily.hpp.

References _mean.

32.70.4.13 void stdair::FareFamily::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Definition at line 96 of file FareFamily.hpp.

References _stdDev.

32.70.4.14 void stdair::FareFamily::setMeanStdDev (const MeanStdDevPairVector_T & *iMeanStdDev*) [inline]

Demand distribution.

Definition at line 99 of file FareFamily.hpp.

References _meanStdDev.

32.70.4.15 void stdair::FareFamily::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 111 of file FareFamily.hpp.

References toString().

32.70.4.16 void stdair::FareFamily::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 120 of file FareFamily.hpp.

32.70.4.17 std::string stdair::FareFamily::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 40 of file FareFamily.cpp.

References describeKey().

Referenced by toStream().

32.70.4.18 const std::string stdair::FareFamily::describeKey () const [inline]

Get a string describing the key.

Definition at line 131 of file FareFamily.hpp.

References _key, and stdair::FareFamilyKey::toString().

Referenced by toString().

32.70.4.19 template<class Archive > void stdair::FareFamily::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 62 of file FareFamily.cpp.

References _key.

32.70.5 Friends And Related Function Documentation

32.70.5.1 friend class FacBom [friend]

Definition at line 29 of file FareFamily.hpp.

32.70.5.2 friend class FacCloneBom [friend]

Definition at line 30 of file FareFamily.hpp.

32.70.5.3 friend class FacBomManager [friend]

Definition at line 31 of file FareFamily.hpp.

32.70.5.4 friend class boost::serialization::access [friend]

Definition at line 32 of file FareFamily.hpp.

32.70.6 Member Data Documentation

32.70.6.1 Key_T stdair::FareFamily::_key

Primary key (fare family code).

Definition at line 184 of file FareFamily.hpp.

Referenced by describeKey(), getFamilyCode(), getKey(), and serialize().

32.70.6.2 BomAbstract* stdair::FareFamily::_parent

Pointer on the parent class (SegmentCabin).

Definition at line 189 of file FareFamily.hpp.

Referenced by getParent().

32.70.6.3 HolderMap_T stdair::FareFamily::_holderMap

Map holding the children (BookingClass objects).

Definition at line 194 of file FareFamily.hpp.

Referenced by getHolderMap().

32.70.6.4 FRAT5Curve_T stdair::FareFamily::_frat5Curve

The associated FRAT5 curve.

Definition at line 199 of file FareFamily.hpp.

Referenced by getFrat5Curve(), and setFrat5Curve().

32.70.6.5 FFDisutilityCurve_T stdair::FareFamily::_disutilityCurve

The associated disutility for the next higher fare family.

Definition at line 204 of file FareFamily.hpp.

Referenced by getDisutilityCurve(), and setDisutilityCurve().

32.70.6.6 MeanValue_T stdair::FareFamily::_mean

Demand distribution forecast.

Definition at line 207 of file FareFamily.hpp.

Referenced by getMean(), and setMean().

32.70.6.7 StdDevValue_T stdair::FareFamily::_stdDev

Definition at line 208 of file FareFamily.hpp.

Referenced by getStdDev(), and setStdDev().

32.70.6.8 MeanStdDevPairVector_T stdair::FareFamily::_meanStdDev

Achievable demand distribution forecast.

Definition at line 213 of file FareFamily.hpp.

Referenced by getMeanStdDev(), and setMeanStdDev().

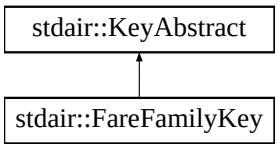
The documentation for this class was generated from the following files:

- stdair/bom/FareFamily.hpp
- stdair/bom/FareFamily.cpp

32.71 stdair::FareFamilyKey Struct Reference

Key of a given fare family, made of a fare family code.

`#include <stdair/bom/FareFamilyKey.hpp>`Inheritance diagram for stdair::FareFamilyKey::



Public Member Functions

- FareFamilyKey (const FamilyCode_T &iFamilyCode)
- FareFamilyKey (const FareFamilyKey &)
- ~FareFamilyKey ()
- const FamilyCode_T & getFamilyCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.71.1 Detailed Description

Key of a given fare family, made of a fare family code.

Definition at line 26 of file FareFamilyKey.hpp.

32.71.2 Constructor & Destructor Documentation

32.71.2.1 stdair::FareFamilyKey::FareFamilyKey (const FamilyCode_T & *iFamilyCode*)

Constructor.

Definition at line 28 of file FareFamilyKey.cpp.

32.71.2.2 stdair::FareFamilyKey::FareFamilyKey (const FareFamilyKey & *iFareFamilyKey*)

Copy constructor.

Definition at line 23 of file FareFamilyKey.cpp.

32.71.2.3 stdair::FareFamilyKey::~FareFamilyKey ()

Destructor.

Definition at line 33 of file FareFamilyKey.cpp.

32.71.3 Member Function Documentation

32.71.3.1 const FamilyCode_T& stdair::FareFamilyKey::getFamilyCode () const [inline]

Get the family code.

Definition at line 56 of file FareFamilyKey.hpp.

Referenced by stdair::FareFamily::getFamilyCode().

32.71.3.2 void stdair::FareFamilyKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file FareFamilyKey.cpp.

References toString().

32.71.3.3 void stdair::FareFamilyKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file FareFamilyKey.cpp.

32.71.3.4 const std::string stdair::FareFamilyKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file FareFamilyKey.cpp.

Referenced by stdair::FareFamily::describeKey(), and toStream().

32.71.3.5 template<class Archive > void stdair::FareFamilyKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 68 of file FareFamilyKey.cpp.

32.71.4 Friends And Related Function Documentation

32.71.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file FareFamilyKey.hpp.

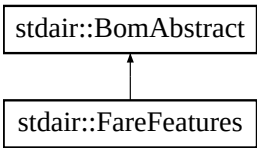
The documentation for this struct was generated from the following files:

- stdair/bom/FareFamilyKey.hpp
- stdair/bom/FareFamilyKey.cpp

32.72 stdair::FareFeatures Class Reference

Class representing the actual attributes for a fare date-period.

`#include <stdair/bom/FareFeatures.hpp>`Inheritance diagram for stdair::FareFeatures::



Public Types

- typedef FareFeaturesKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const TripType_T & getTripType () const
- const DayDuration_T & getAdvancePurchase () const
- const SaturdayStay_T & getSaturdayStay () const
- const ChangeFees_T & getChangeFees () const
- const NonRefundable_T & getRefundableOption () const
- const DayDuration_T & getMinimumStay () const
- bool isTripTypeValid (const TripType_T &) const
- bool isStayDurationValid (const DayDuration_T &) const
- bool isAdvancePurchaseValid (const DateTime_T &iBookingRequestDateTime, const DateTime_T &iFlightDateTime) const

Protected Member Functions

- FareFeatures (const Key_T &)
- virtual ~FareFeatures ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.72.1 Detailed Description

Class representing the actual attributes for a fare date-period.

Definition at line 18 of file FareFeatures.hpp.

32.72.2 Member Typedef Documentation

32.72.2.1 typedef FareFeaturesKey stdair::FareFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file FareFeatures.hpp.

32.72.3 Constructor & Destructor Documentation

32.72.3.1 stdair::FareFeatures::FareFeatures (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 33 of file FareFeatures.cpp.

32.72.3.2 stdair::FareFeatures::~FareFeatures () `[protected, virtual]`

Destructor.

Definition at line 38 of file FareFeatures.cpp.

32.72.4 Member Function Documentation

32.72.4.1 void stdair::FareFeatures::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 37 of file FareFeatures.hpp.

References toString().

32.72.4.2 void stdair::FareFeatures::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 46 of file FareFeatures.hpp.

32.72.4.3 std::string stdair::FareFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file FareFeatures.cpp.

References describeKey().

Referenced by toStream().

32.72.4.4 const std::string stdair::FareFeatures::describeKey () const [inline]

Get a string describing the key.

Definition at line 57 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::toString().

Referenced by toString().

32.72.4.5 const Key_T& stdair::FareFeatures::getKey () const [inline]

Get the primary key (trip type, advance purchase,... ,cabin code).

Definition at line 67 of file FareFeatures.hpp.

References _key.

32.72.4.6 BomAbstract* const stdair::FareFeatures::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file FareFeatures.hpp.

References _parent.

32.72.4.7 const HolderMap_T& stdair::FareFeatures::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file FareFeatures.hpp.

References _holderMap.

32.72.4.8 const TripType_T& stdair::FareFeatures::getTripType () const [inline]

Get the trip type.

Definition at line 88 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getTripType().

Referenced by isTripTypeValid().

32.72.4.9 const DayDuration_T& stdair::FareFeatures::getAdvancePurchase () const [inline]

Get the fare day duration.

Definition at line 95 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getAdvancePurchase().

Referenced by isAdvancePurchaseValid().

32.72.4.10 const SaturdayStay_T& stdair::FareFeatures::getSaturdayStay () const [inline]

Get the fare saturday stay option.

Definition at line 102 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getSaturdayStay().

32.72.4.11 const ChangeFees_T& stdair::FareFeatures::getChangeFees () const [inline]

Get the change fees criterion.

Definition at line 109 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getChangeFees().

32.72.4.12 const NonRefundable_T& stdair::FareFeatures::getRefundableOption () const [inline]

Get the refundable option.

Definition at line 116 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getRefundableOption().

32.72.4.13 const DayDuration_T& stdair::FareFeatures::getMinimumStay () const [inline]

Get the minimum stay.

Definition at line 123 of file FareFeatures.hpp.

References _key, and stdair::FareFeaturesKey::getMinimumStay().

Referenced by isStayDurationValid().

32.72.4.14 bool stdair::FareFeatures::isTripTypeValid (const TripType_T & *iBookingRequestTripType*) const

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 50 of file FareFeatures.cpp.

References getTripType(), stdair::TRIP_TYPE_INBOUND, stdair::TRIP_TYPE_OUTBOUND, and stdair::TRIP_TYPE_ROUND_TRIP.

32.72.4.15 bool stdair::FareFeatures::isStayDurationValid (const DayDuration_T & *iStayDuration*) const

Check whether a given stay duration is greater or equal to the minimum stay of the fare rule.

Definition at line 75 of file FareFeatures.cpp.

References getMinimumStay().

32.72.4.16 bool stdair::FareFeatures::isAdvancePurchaseValid (const DateTime_T & *iBookingRequestDateTime*, const DateTime_T & *iFlightDateTime*) const

Check whether a booking request date is valid compared the required advance purchase number of days of the fare rule.

Definition at line 88 of file FareFeatures.cpp.

References getAdvancePurchase().

32.72.5 Friends And Related Function Documentation

32.72.5.1 friend class FacBom [friend]

Definition at line 19 of file FareFeatures.hpp.

32.72.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file FareFeatures.hpp.

32.72.5.3 friend class FacBomManager [friend]

Definition at line 21 of file FareFeatures.hpp.

32.72.6 Member Data Documentation

32.72.6.1 Key_T stdair::FareFeatures::_key [protected]

Primary key (flight number and departure date).

Definition at line 176 of file FareFeatures.hpp.

Referenced by describeKey(), getAdvancePurchase(), getChangeFees(), getKey(), getMinimumStay(), getRefundableOption(), getSaturdayStay(), and getTripType().

32.72.6.2 BomAbstract* stdair::FareFeatures::_parent [protected]

Pointer on the parent class.

Definition at line 181 of file FareFeatures.hpp.

Referenced by getParent().

32.72.6.3 HolderMap_T stdair::FareFeatures::_holderMap [protected]

Map holding the children.

Definition at line 186 of file FareFeatures.hpp.

Referenced by getHolderMap().

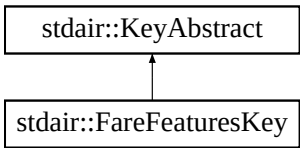
The documentation for this class was generated from the following files:

- stdair/bom/FareFeatures.hpp
- stdair/bom/FareFeatures.cpp

32.73 stdair::FareFeaturesKey Struct Reference

Key of date-period.

`#include <stdair/bom/FareFeaturesKey.hpp>`Inheritance diagram for stdair::FareFeaturesKey::



Public Member Functions

- FareFeaturesKey (const TripType_T &, const DayDuration_T &, const SaturdayStay_T &, const ChangeFees_T &, const NonRefundable_T &, const DayDuration_T &)
- FareFeaturesKey (const FareFeaturesKey &)
- ~FareFeaturesKey ()
- const TripType_T & getTripType () const
- const DayDuration_T & getAdvancePurchase () const
- const SaturdayStay_T & getSaturdayStay () const

- const ChangeFees_T & getChangeFees () const
- const NonRefundable_T & getRefundableOption () const
- const DayDuration_T & getMinimumStay () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.73.1 Detailed Description

Key of date-period.

Definition at line 18 of file FareFeaturesKey.hpp.

32.73.2 Constructor & Destructor Documentation

32.73.2.1 stdair::FareFeaturesKey::FareFeaturesKey (const TripType_T & *iTripType*, const DayDuration_T & *iAdvancePurchase*, const SaturdayStay_T & *iSaturdayStay*, const ChangeFees_T & *iChangeFees*, const NonRefundable_T & *iNonRefundable*, const DayDuration_T & *iMinimumStay*)

Main constructor.

Definition at line 26 of file FareFeaturesKey.cpp.

32.73.2.2 stdair::FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey & *iKey*)

Copy constructor.

Definition at line 38 of file FareFeaturesKey.cpp.

32.73.2.3 stdair::FareFeaturesKey::~FareFeaturesKey ()

Destructor.

Definition at line 48 of file FareFeaturesKey.cpp.

32.73.3 Member Function Documentation

32.73.3.1 const TripType_T& stdair::FareFeaturesKey::getTripType () const `[inline]`

Get the fare trip type.

Definition at line 39 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getTripType().

32.73.3.2 const DayDuration_T& stdair::FareFeaturesKey::getAdvancePurchase () const `[inline]`

Get the fare day duration.

Definition at line 46 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getAdvancePurchase().

32.73.3.3 const SaturdayStay_T& stdair::FareFeaturesKey::getSaturdayStay () const [inline]

Get the fare saturday stay option.

Definition at line 53 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getSaturdayStay().

32.73.3.4 const ChangeFees_T& stdair::FareFeaturesKey::getChangeFees () const [inline]

Get the change fees criterion.

Definition at line 60 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getChangeFees().

32.73.3.5 const NonRefundable_T& stdair::FareFeaturesKey::getRefundableOption () const [inline]

Get the refundable option.

Definition at line 67 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getRefundableOption().

32.73.3.6 const DayDuration_T& stdair::FareFeaturesKey::getMinimumStay () const [inline]

Get the minimum stay.

Definition at line 74 of file FareFeaturesKey.hpp.

Referenced by stdair::FareFeatures::getMinimumStay().

32.73.3.7 void stdair::FareFeaturesKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 52 of file FareFeaturesKey.cpp.

References toString().

32.73.3.8 void stdair::FareFeaturesKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 57 of file FareFeaturesKey.cpp.

32.73.3.9 const std::string stdair::FareFeaturesKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 61 of file FareFeaturesKey.cpp.

Referenced by stdair::FareFeatures::describeKey(), and toStream().

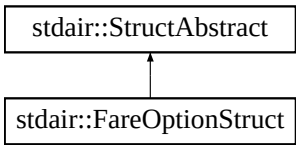
The documentation for this struct was generated from the following files:

- stdair/bom/FareFeaturesKey.hpp
- stdair/bom/FareFeaturesKey.cpp

32.74 stdair::FareOptionStruct Struct Reference

Structure holding the elements of a fare option.

`#include <stdair/bom/FareOptionStruct.hpp>`Inheritance diagram for stdair::FareOptionStruct::



Public Member Functions

- const ClassList_StringList_T & getClassPath () const
- const Fare_T & getFare () const
- const Availability_T & getAvailability () const
- const ChangeFees_T getChangeFees () const
- const NonRefundable_T getNonRefundable () const
- const SaturdayStay_T getSaturdayStay () const
- void addClassList (const std::string)
- void emptyClassList ()
- void setFare (const Fare_T &iFare)
- void setAvailability (const Availability_T &iAvl)
- void setChangeFees (const ChangeFees_T iRes)
- void setNonRefundable (const NonRefundable_T iRes)
- void setSaturdayStay (const SaturdayStay_T iRes)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const

- FareOptionStruct ()
- FareOptionStruct (const std::string &iClassPath, const Fare_T &, const ChangeFees_T &, const NonRefundable_T &, const SaturdayStay_T &)
- FareOptionStruct (const FareOptionStruct &)
- ~FareOptionStruct ()

32.74.1 Detailed Description

Structure holding the elements of a fare option.

Definition at line 20 of file FareOptionStruct.hpp.

32.74.2 Constructor & Destructor Documentation

32.74.2.1 stdair::FareOptionStruct::FareOptionStruct ()

Default constructor.

Definition at line 14 of file FareOptionStruct.cpp.

32.74.2.2 stdair::FareOptionStruct::FareOptionStruct (const std::string & *iClassPath*, const Fare_T & *iFare*, const ChangeFees_T & *iChangeFee*, const NonRefundable_T & *iNonRefundable*, const SaturdayStay_T & *iSaturdayNightStay*)

Main constructor.

Definition at line 26 of file FareOptionStruct.cpp.

32.74.2.3 stdair::FareOptionStruct::FareOptionStruct (const FareOptionStruct & *iFO*)

Copy constructor.

Definition at line 19 of file FareOptionStruct.cpp.

32.74.2.4 stdair::FareOptionStruct::~FareOptionStruct ()

Destructor.

Definition at line 38 of file FareOptionStruct.cpp.

32.74.3 Member Function Documentation

32.74.3.1 const ClassList_StringList_T& stdair::FareOptionStruct::getClassPath () const `[inline]`

Get the class-path.

Definition at line 24 of file FareOptionStruct.hpp.

32.74.3.2 const Fare_T& stdair::FareOptionStruct::getFare () const [inline]

Get the fare value.

Definition at line 29 of file FareOptionStruct.hpp.

32.74.3.3 const Availability_T& stdair::FareOptionStruct::getAvailability () const [inline]

Get the availability.

Definition at line 34 of file FareOptionStruct.hpp.

32.74.3.4 const ChangeFees_T stdair::FareOptionStruct::getChangeFees () const [inline]

Get the change fees.

Definition at line 39 of file FareOptionStruct.hpp.

32.74.3.5 const NonRefundable_T stdair::FareOptionStruct::getNonRefundable () const [inline]

State whether the ticket is refundable.

Definition at line 44 of file FareOptionStruct.hpp.

32.74.3.6 const SaturdayStay_T stdair::FareOptionStruct::getSaturdayStay () const [inline]

State whether there is a condition on the saturday night stay.

Definition at line 49 of file FareOptionStruct.hpp.

32.74.3.7 void stdair::FareOptionStruct::addClassList (const std::string *iClassCodeList*)

Set the class-path.

Definition at line 93 of file FareOptionStruct.cpp.

32.74.3.8 void stdair::FareOptionStruct::emptyClassList ()

Empty the class-path.

Definition at line 98 of file FareOptionStruct.cpp.

32.74.3.9 void stdair::FareOptionStruct::setFare (const Fare_T & *iFare*) [inline]

Set the fare value.

Definition at line 63 of file FareOptionStruct.hpp.

32.74.3.10 void stdair::FareOptionStruct::setAvailability (const Availability_T & *iAvl*) [inline]

Set the availability.

Definition at line 68 of file FareOptionStruct.hpp.

32.74.3.11 void stdair::FareOptionStruct::setChangeFees (const ChangeFees_T *iRes*) [inline]

Set the change fees.

Definition at line 73 of file FareOptionStruct.hpp.

32.74.3.12 void stdair::FareOptionStruct::setNonRefundable (const NonRefundable_T *iRes*) [inline]

Set the flag for the ticket refundability.

Definition at line 78 of file FareOptionStruct.hpp.

32.74.3.13 void stdair::FareOptionStruct::setSaturdayStay (const SaturdayStay_T *iRes*) [inline]

Set the flag for the saturday night stay condition.

Definition at line 83 of file FareOptionStruct.hpp.

32.74.3.14 void stdair::FareOptionStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 42 of file FareOptionStruct.cpp.

References describe().

32.74.3.15 void stdair::FareOptionStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 47 of file FareOptionStruct.cpp.

32.74.3.16 const std::string stdair::FareOptionStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 51 of file FareOptionStruct.cpp.

Referenced by stdair::TravelSolutionStruct::describe(), and toStream().

32.74.3.17 const std::string stdair::FareOptionStruct::display () const

Display of the structure.

Definition at line 73 of file FareOptionStruct.cpp.

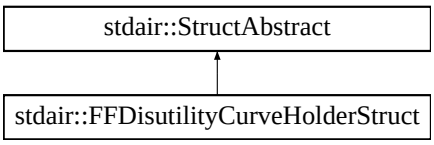
Referenced by stdair::TravelSolutionStruct::display().

The documentation for this struct was generated from the following files:

- stdair/bom/FareOptionStruct.hpp
- stdair/bom/FareOptionStruct.cpp

32.75 stdair::FFDisutilityCurveHolderStruct Struct Reference

`#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>`Inheritance diagram for stdair::FFDisutilityCurveHolderStruct::



Public Member Functions

- const FFDisutilityCurve_T & getFFDisutilityCurve (const std::string &) const
- void addCurve (const std::string &, const FFDisutilityCurve_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- FFDisutilityCurveHolderStruct ()
- FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct &)
- ~FFDisutilityCurveHolderStruct ()

32.75.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file FFDisutilityCurveHolderStruct.hpp.

32.75.2 Constructor & Destructor Documentation

32.75.2.1 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct ()

Constructor.

Definition at line 14 of file FFDisutilityCurveHolderStruct.cpp.

32.75.2.2 stdair::FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct & *iHolder*)

Copy constructor.

Definition at line 19 of file FFDisutilityCurveHolderStruct.cpp.

32.75.2.3 stdair::FFDisutilityCurveHolderStruct::~FFDisutilityCurveHolderStruct ()

Destructor.

Definition at line 24 of file FFDisutilityCurveHolderStruct.cpp.

32.75.3 Member Function Documentation

32.75.3.1 const FFDisutilityCurve_T & stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve (const std::string & *iKey*) const

Get the FFDisutility curve corresponding to the given key.

Definition at line 29 of file FFDisutilityCurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::getFFDisutilityCurve().

32.75.3.2 void stdair::FFDisutilityCurveHolderStruct::addCurve (const std::string & *iKey*, const FFDisutilityCurve_T & *iCurve*)

Add a new curve to the holder.

Definition at line 42 of file FFDisutilityCurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::addFFDisutilityCurve().

32.75.3.3 void stdair::FFDisutilityCurveHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 53 of file FFDisutilityCurveHolderStruct.cpp.

References describe().

32.75.3.4 void stdair::FFDisutilityCurveHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file FFDisutilityCurveHolderStruct.cpp.

32.75.3.5 const std::string stdair::FFDisutilityCurveHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 62 of file FFDisutilityCurveHolderStruct.cpp.

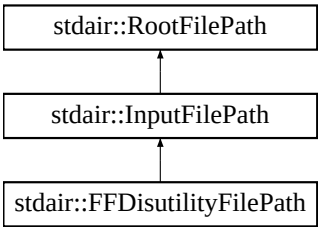
Referenced by toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/FFDisutilityCurveHolderStruct.hpp
- stdair/bom/FFDisutilityCurveHolderStruct.cpp

32.76 stdair::FFDisutilityFilePath Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::FFDisutilityFilePath::



Public Member Functions

- FFDisutilityFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.76.1 Detailed Description

FFDisutility input file.

Definition at line 100 of file stdair_file.hpp.

32.76.2 Constructor & Destructor Documentation

32.76.2.1 stdair::FFDisutilityFilePath::FFDisutilityFilePath (const Filename_T & *iFilename*) [inline, explicit]

Constructor.

Definition at line 105 of file stdair_file.hpp.

32.76.3 Member Function Documentation

32.76.3.1 const char* stdair::RootFilePath::name () const [inline, inherited]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.76.4 Member Data Documentation

32.76.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

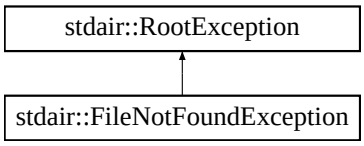
Referenced by stdair::RootFilePath::name().

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

- stdair/stdair_file.hpp

32.77 stdair::FileNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::FileNotFoundException::



Public Member Functions

- FileNotFoundException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.77.1 Detailed Description

File not found.

Definition at line 50 of file stdair_exceptions.hpp.

32.77.2 Constructor & Destructor Documentation

32.77.2.1 stdair::FileNotFoundException::FileNotFoundException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 53 of file stdair_exceptions.hpp.

32.77.3 Member Function Documentation

32.77.3.1 const char∗ stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.77.4 Member Data Documentation

32.77.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

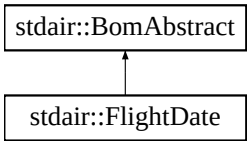
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

32.78 stdair::FlightDate Class Reference

Class representing the actual attributes for an airline flight-date.

`#include <stdair/bom/FlightDate.hpp>`Inheritance diagram for stdair::FlightDate::



Public Types

- typedef FlightDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FlightNumber_T & getFlightNumber () const
- const Date_T & getDepartureDate () const
- const AirlineCode_T & getAirlineCode () const
- const HolderMap_T & getHolderMap () const
- LegDate * getLegDate (const std::string &iLegDateKeyStr) const
- LegDate * getLegDate (const LegDateKey &) const
- SegmentDate * getSegmentDate (const std::string &iSegmentDateKeyStr) const
- SegmentDate * getSegmentDate (const SegmentDateKey &) const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- FlightDate (const Key_T &)
- virtual ~FlightDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.78.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 35 of file FlightDate.hpp.

32.78.2 Member Typedef Documentation

32.78.2.1 typedef FlightDateKey stdair::FlightDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 46 of file FlightDate.hpp.

32.78.3 Constructor & Destructor Documentation

32.78.3.1 stdair::FlightDate::FlightDate (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 29 of file FlightDate.cpp.

32.78.3.2 stdair::FlightDate::~FlightDate () `[protected, virtual]`

Destructor.

Definition at line 33 of file FlightDate.cpp.

32.78.4 Member Function Documentation

32.78.4.1 const Key_T& stdair::FlightDate::getKey () const `[inline]`

Get the flight-date key.

Definition at line 52 of file FlightDate.hpp.

References _key.

32.78.4.2 BomAbstract* const stdair::FlightDate::getParent () const `[inline]`

Get the parent object.

Definition at line 57 of file FlightDate.hpp.

References _parent.

Referenced by getAirlineCode().

32.78.4.3 const FlightNumber_T& stdair::FlightDate::getFlightNumber () const `[inline]`

Get the flight number (part of the primary key).

Definition at line 62 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::getFlightNumber().

Referenced by stdair::BomJSONExport::jsonExportFlightDateList(), and stdair::BomJSONExport::jsonExportFlightDateObjects().

32.78.4.4 const Date_T& stdair::FlightDate::getDepartureDate () const [inline]

Get the flight date (part of the primary key).

Definition at line 67 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::getDepartureDate().

Referenced by stdair::LegDate::describeRoutingKey(), stdair::BomJSONExport::jsonExportFlightDateList(), and stdair::BomJSONExport::jsonExportFlightDateObjects().

32.78.4.5 const AirlineCode_T & stdair::FlightDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 37 of file FlightDate.cpp.

References stdair::Inventory::getAirlineCode(), and getParent().

Referenced by stdair::LegDate::getAirlineCode(), and stdair::BomJSONExport::jsonExportFlightDateObjects().

32.78.4.6 const HolderMap_T& stdair::FlightDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 83 of file FlightDate.hpp.

References _holderMap.

32.78.4.7 LegDate * stdair::FlightDate::getLegDate (const std::string & *iLegDateKeyStr*) const

Get a pointer on the LegDate object corresponding to the given key.

Note:

The LegDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The leg-date key.

Returns:

LegDate* Found LegDate object. NULL if not found.

Definition at line 52 of file FlightDate.cpp.

Referenced by getLegDate(), stdair::BomRetriever::retrieveDummyLegCabin(), and stdair::BomRetriever::retrieveOperatingLegDateFromLongKey().

32.78.4.8 LegDate ∗ stdair::FlightDate::getLegDate (const LegDateKey & *iLegDateKey*) const

Get a pointer on the LegDate object corresponding to the given key.

Note:

The LegDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const LegDateKey& The leg-date key

Returns:

LegDate∗ Found LegDate object. NULL if not found.

Definition at line 59 of file FlightDate.cpp.

References getLegDate(), and stdair::LegDateKey::toString().

32.78.4.9 SegmentDate ∗ stdair::FlightDate::getSegmentDate (const std::string & *iSegmentDateKeyStr*) const

Get a pointer on the SegmentDate object corresponding to the given key.

Note:

The SegmentDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The segment-date key.

Returns:

SegmentDate∗ Found SegmentDate object. NULL if not found.

Definition at line 65 of file FlightDate.cpp.

Referenced by getSegmentDate(), stdair::BomRetriever::retrieveDummySegmentCabin(), stdair::BomRetriever::retrieveSegmentDateFromKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.78.4.10 SegmentDate ∗ stdair::FlightDate::getSegmentDate (const SegmentDateKey & *iSegmentDateKey*) const

Get a pointer on the SegmentDate object corresponding to the given key.

Note:

The SegmentDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const SegmentDateKey& The segment-date key

Returns:

SegmentDate∗ Found SegmentDate object. NULL if not found.

Definition at line 73 of file FlightDate.cpp.

References getSegmentDate(), and stdair::SegmentDateKey::toString().

32.78.4.11 void stdair::FlightDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 142 of file FlightDate.hpp.

References toString().

32.78.4.12 void stdair::FlightDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 151 of file FlightDate.hpp.

32.78.4.13 std::string stdair::FlightDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 45 of file FlightDate.cpp.

References describeKey().

Referenced by toStream().

32.78.4.14 const std::string stdair::FlightDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 162 of file FlightDate.hpp.

References _key, and stdair::FlightDateKey::toString().

Referenced by toString().

32.78.4.15 template<class Archive > void stdair::FlightDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 187 of file CmdBomSerialiser.cpp.

References _key.

32.78.5 Friends And Related Function Documentation

32.78.5.1 friend class FacBom [friend]

Definition at line 36 of file FlightDate.hpp.

32.78.5.2 friend class FacCloneBom [friend]

Definition at line 37 of file FlightDate.hpp.

32.78.5.3 friend class FacBomManager [friend]

Definition at line 38 of file FlightDate.hpp.

32.78.5.4 friend class boost::serialization::access [friend]

Definition at line 39 of file FlightDate.hpp.

32.78.6 Member Data Documentation

32.78.6.1 Key_T stdair::FlightDate::_key [protected]

Primary key (flight number and departure date).

Definition at line 216 of file FlightDate.hpp.

Referenced by describeKey(), getDepartureDate(), getFlightNumber(), getKey(), and serialize().

32.78.6.2 BomAbstract* stdair::FlightDate::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 221 of file FlightDate.hpp.

Referenced by getParent().

32.78.6.3 HolderMap_T stdair::FlightDate::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 226 of file FlightDate.hpp.

Referenced by getHolderMap().

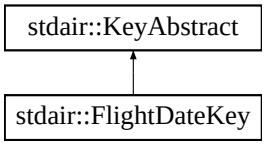
The documentation for this class was generated from the following files:

- stdair/bom/FlightDate.hpp
- stdair/bom/FlightDate.cpp
- stdair/command/CmdBomSerialiser.cpp

32.79 stdair::FlightDateKey Struct Reference

Key of a given flight-date, made of a flight number and a departure date.

`#include <stdair/bom/FlightDateKey.hpp>`Inheritance diagram for stdair::FlightDateKey::



Public Member Functions

- FlightDateKey (const FlightNumber_T &, const Date_T &)
- FlightDateKey (const FlightDateKey &)
- ~FlightDateKey ()
- const FlightNumber_T & getFlightNumber () const
- const Date_T & getDepartureDate () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.79.1 Detailed Description

Key of a given flight-date, made of a flight number and a departure date.

Definition at line 28 of file FlightDateKey.hpp.

32.79.2 Constructor & Destructor Documentation

32.79.2.1 stdair::FlightDateKey::FlightDateKey (const FlightNumber_T & *iFlightNumber*, const Date_T & *iFlightDate*)

Constructor.

Definition at line 28 of file FlightDateKey.cpp.

32.79.2.2 stdair::FlightDateKey::FlightDateKey (const FlightDateKey & *iKey*)

Copy constructor.

Definition at line 34 of file FlightDateKey.cpp.

32.79.2.3 stdair::FlightDateKey::~FlightDateKey ()

Destructor.

Definition at line 39 of file FlightDateKey.cpp.

32.79.3 Member Function Documentation

32.79.3.1 const FlightNumber_T& stdair::FlightDateKey::getFlightNumber () const `[inline]`

Get the flight number.

Definition at line 58 of file FlightDateKey.hpp.

Referenced by stdair::FlightDate::getFlightNumber().

32.79.3.2 const Date_T& stdair::FlightDateKey::getDepartureDate () const `[inline]`

Get the departure date of the (first leg of the) flight.

Definition at line 63 of file FlightDateKey.hpp.

Referenced by stdair::OnDDateKey::getDate(), and stdair::FlightDate::getDepartureDate().

32.79.3.3 void stdair::FlightDateKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file FlightDateKey.cpp.

References toString().

32.79.3.4 void stdair::FlightDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 48 of file FlightDateKey.cpp.

32.79.3.5 const std::string stdair::FlightDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 52 of file FlightDateKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::FlightDate::describeKey(), stdair::Inventory::getFlightDate(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toStream().

32.79.3.6 template<class Archive > void stdair::FlightDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 77 of file FlightDateKey.cpp.

32.79.4 Friends And Related Function Documentation

32.79.4.1 friend class boost::serialization::access [friend]

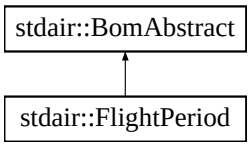
Definition at line 29 of file FlightDateKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/FlightDateKey.hpp
- stdair/bom/FlightDateKey.cpp

32.80 stdair::FlightPeriod Class Reference

`#include <stdair/bom/FlightPeriod.hpp>`Inheritance diagram for stdair::FlightPeriod::



Public Types

- typedef FlightPeriodKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const FlightNumber_T & getFlightNumber () const
- const PeriodStruct & getPeriod () const
- const HolderMap_T & getHolderMap () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Protected Member Functions

- FlightPeriod (const Key_T &)
- ~FlightPeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.80.1 Detailed Description

Class representing the actual attributes for an airline flight-period.

Definition at line 15 of file FlightPeriod.hpp.

32.80.2 Member Typedef Documentation

32.80.2.1 typedef FlightPeriodKey stdair::FlightPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file FlightPeriod.hpp.

32.80.3 Constructor & Destructor Documentation

32.80.3.1 stdair::FlightPeriod::FlightPeriod (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 12 of file FlightPeriod.cpp.

32.80.3.2 stdair::FlightPeriod::~FlightPeriod () [protected]

Destructor.

Definition at line 22 of file FlightPeriod.cpp.

32.80.4 Member Function Documentation

32.80.4.1 const Key_T& stdair::FlightPeriod::getKey () const [inline]

Get the flight-period key.

Definition at line 28 of file FlightPeriod.hpp.

References _key.

32.80.4.2 BomAbstract* const stdair::FlightPeriod::getParent () const [inline]

Get the parent object.

Definition at line 31 of file FlightPeriod.hpp.

References _parent.

32.80.4.3 const FlightNumber_T& stdair::FlightPeriod::getFlightNumber () const [inline]

Get the flight number (part of the primary key).

Definition at line 34 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::getFlightNumber().

32.80.4.4 const PeriodStruct& stdair::FlightPeriod::getPeriod () const [inline]

Get the departure period (part of the key).

Definition at line 39 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::getPeriod().

32.80.4.5 const HolderMap_T& stdair::FlightPeriod::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 42 of file FlightPeriod.hpp.

References _holderMap.

32.80.4.6 void stdair::FlightPeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 49 of file FlightPeriod.hpp.

References toString().

32.80.4.7 void stdair::FlightPeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 53 of file FlightPeriod.hpp.

32.80.4.8 std::string stdair::FlightPeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 26 of file FlightPeriod.cpp.

References describeKey().

Referenced by toStream().

32.80.4.9 const std::string stdair::FlightPeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 59 of file FlightPeriod.hpp.

References _key, and stdair::FlightPeriodKey::toString().

Referenced by toString().

32.80.5 Friends And Related Function Documentation

32.80.5.1 friend class FacBom [friend]

Definition at line 16 of file FlightPeriod.hpp.

32.80.5.2 friend class FacCloneBom [friend]

Definition at line 17 of file FlightPeriod.hpp.

32.80.5.3 friend class FacBomManager [friend]

Definition at line 18 of file FlightPeriod.hpp.

32.80.6 Member Data Documentation

32.80.6.1 Key_T stdair::FlightPeriod::_key [protected]

Definition at line 86 of file FlightPeriod.hpp.

Referenced by describeKey(), getFlightNumber(), getKey(), and getPeriod().

32.80.6.2 BomAbstract* stdair::FlightPeriod::_parent [protected]

Definition at line 87 of file FlightPeriod.hpp.

Referenced by getParent().

32.80.6.3 HolderMap_T stdair::FlightPeriod::_holderMap [protected]

Definition at line 88 of file FlightPeriod.hpp.

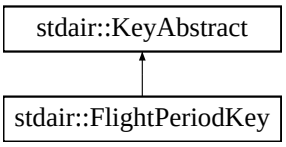
Referenced by getHolderMap().

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

- stdair/bom/FlightPeriod.hpp
- stdair/bom/FlightPeriod.cpp

32.81 stdair::FlightPeriodKey Struct Reference

`#include <stdair/bom/FlightPeriodKey.hpp>`Inheritance diagram for stdair::FlightPeriodKey::



Public Member Functions

- FlightPeriodKey (const FlightNumber_T &, const PeriodStruct &)
- FlightPeriodKey (const FlightPeriodKey &)
- ~FlightPeriodKey ()
- const FlightNumber_T & getFlightNumber () const
- const PeriodStruct & getPeriod () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.81.1 Detailed Description

Key of flight-period.

Definition at line 13 of file FlightPeriodKey.hpp.

32.81.2 Constructor & Destructor Documentation

32.81.2.1 stdair::FlightPeriodKey::FlightPeriodKey (const FlightNumber_T & *iFlightNumber*, const PeriodStruct & *iPeriod*)

Constructors.

Definition at line 10 of file FlightPeriodKey.cpp.

32.81.2.2 stdair::FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey & *iKey*)

Definition at line 16 of file FlightPeriodKey.cpp.

32.81.2.3 stdair::FlightPeriodKey::~FlightPeriodKey ()

Destructor.

Definition at line 21 of file FlightPeriodKey.cpp.

32.81.3 Member Function Documentation

32.81.3.1 const FlightNumber_T& stdair::FlightPeriodKey::getFlightNumber () const `[inline]`

Get the flight number.

Definition at line 28 of file FlightPeriodKey.hpp.

Referenced by stdair::FlightPeriod::getFlightNumber().

32.81.3.2 const PeriodStruct& stdair::FlightPeriodKey::getPeriod () const [inline]

Get the active days-of-week.

Definition at line 33 of file FlightPeriodKey.hpp.

Referenced by stdair::FlightPeriod::getPeriod().

32.81.3.3 void stdair::FlightPeriodKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 25 of file FlightPeriodKey.cpp.

References toString().

32.81.3.4 void stdair::FlightPeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 30 of file FlightPeriodKey.cpp.

32.81.3.5 const std::string stdair::FlightPeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from stdair::KeyAbstract.

Definition at line 34 of file FlightPeriodKey.cpp.

References stdair::PeriodStruct::describeShort().

Referenced by stdair::FlightPeriod::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/FlightPeriodKey.hpp
- stdair/bom/FlightPeriodKey.cpp

32.82 FloatingPoint< RawType > Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef TypeWithSize< sizeof(RawType)>::UInt Bits

Public Member Functions

- FloatingPoint (const RawType &x)
- const Bits & bits () const
- Bits exponent_bits () const
- Bits fraction_bits () const
- Bits sign_bit () const
- bool is_nan () const
- bool AlmostEquals (const FloatingPoint &rhs) const

Static Public Member Functions

- static RawType ReinterpretBits (const Bits bits)
- static RawType Infinity ()

Static Public Attributes

- static const size_t kBitCount = 8∗sizeof(RawType)
- static const size_t kFractionBitCount
- static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount
- static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1)
- static const Bits kFractionBitMask
- static const Bits kExponentBitMask = ∼(kSignBitMask | kFractionBitMask)
- static const size_t kMaxUlps = 4

32.82.1 Detailed Description

template<typename RawType> class FloatingPoint< RawType >

Definition at line 117 of file float_utils_google.hpp.

32.82.2 Member Typedef Documentation

32.82.2.1 template<typename RawType> typedef TypeWithSize<sizeof(RawType)>::UInt FloatingPoint< RawType >::Bits

Definition at line 121 of file float_utils_google.hpp.

32.82.3 Constructor & Destructor Documentation

32.82.3.1 template<typename RawType> FloatingPoint< RawType >::FloatingPoint (const RawType & *x*) [inline, explicit]

Definition at line 165 of file float_utils_google.hpp.

32.82.4 Member Function Documentation

32.82.4.1 template<typename RawType> static RawType FloatingPoint< RawType >::ReinterpretBits (const Bits *bits*) [inline, static]

Definition at line 172 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::Infinity().

32.82.4.2 template<typename RawType> static RawType FloatingPoint< RawType >::Infinity () [inline, static]

Definition at line 179 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask, and FloatingPoint< RawType >::ReinterpretBits().

32.82.4.3 template<typename RawType> const Bits& FloatingPoint< RawType >::bits () const [inline]

Definition at line 186 of file float_utils_google.hpp.

32.82.4.4 template<typename RawType> Bits FloatingPoint< RawType >::exponent_bits () const [inline]

Definition at line 189 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

32.82.4.5 template<typename RawType> Bits FloatingPoint< RawType >::fraction_bits () const [inline]

Definition at line 192 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kFractionBitMask.

Referenced by FloatingPoint< RawType >::is_nan().

32.82.4.6 template<typename RawType> Bits FloatingPoint< RawType >::sign_bit () const [inline]

Definition at line 195 of file float_utils_google.hpp.

References FloatingPoint< RawType >::kSignBitMask.

32.82.4.7 template<typename RawType> bool FloatingPoint< RawType >::is_nan () const [inline]

Definition at line 198 of file float_utils_google.hpp.

References FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::fraction_bits(), and FloatingPoint< RawType >::kExponentBitMask.

Referenced by FloatingPoint< RawType >::AlmostEquals().

32.82.4.8 template<typename RawType> bool FloatingPoint< RawType >::AlmostEquals (const FloatingPoint< RawType > & *rhs*) const [inline]

Definition at line 210 of file float_utils_google.hpp.

References FloatingPoint< RawType >::is_nan(), and FloatingPoint< RawType >::kMaxUlps.

32.82.5 Member Data Documentation

32.82.5.1 template<typename RawType> const size_t FloatingPoint< RawType >::kBitCount = 8∗sizeof(RawType) [static]

Definition at line 126 of file float_utils_google.hpp.

32.82.5.2 template<typename RawType> const size_t FloatingPoint< RawType >::kFractionBitCount [static]

Initial value:

```
std::numeric_limits<RawType>::digits - 1
```

Definition at line 129 of file float_utils_google.hpp.

32.82.5.3 template<typename RawType> const size_t FloatingPoint< RawType >::kExponentBitCount = kBitCount - 1 - kFractionBitCount [static]

Definition at line 133 of file float_utils_google.hpp.

32.82.5.4 template<typename RawType> const Bits FloatingPoint< RawType >::kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1) [static]

Definition at line 136 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::sign_bit().

32.82.5.5 template<typename RawType> const Bits FloatingPoint< RawType >::kFractionBitMask [static]

Initial value:

```
~static_cast<Bits>(0) >> (kExponentBitCount + 1)
```

Definition at line 139 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::fraction_bits().

32.82.5.6 template<typename RawType> const Bits FloatingPoint< RawType >::kExponentBitMask = ~(kSignBitMask | kFractionBitMask) [static]

Definition at line 143 of file float_utils_google.hpp.

Referenced by FloatingPoint< RawType >::exponent_bits(), FloatingPoint< RawType >::Infinity(), and FloatingPoint< RawType >::is_nan().

32.82.5.7 template<typename RawType> const size_t FloatingPoint< RawType >::kMaxUlps = 4 [static]

Definition at line 157 of file float_utils_google.hpp.

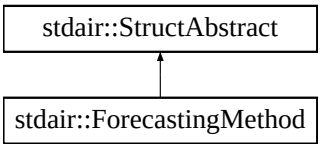
Referenced by FloatingPoint< RawType >::AlmostEquals().

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

- stdair/basic/float_utils_google.hpp

32.83 stdair::ForecastingMethod Struct Reference

`#include <stdair/basic/ForecastingMethod.hpp>`Inheritance diagram for stdair::ForecastingMethod::



Public Types

- enum EN_ForecastingMethod {
  Q_FORECASTING = 0, HYBRID_FORECASTING, OLD_QFF, NEW_QFF,
  BASED_FORECASTING, LAST_VALUE }

Public Member Functions

- EN_ForecastingMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_ForecastingMethod &) const
- ForecastingMethod (const EN_ForecastingMethod &)
- ForecastingMethod (const char iMethod)
- ForecastingMethod (const ForecastingMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_ForecastingMethod &)
- static char getMethodLabel (const EN_ForecastingMethod &)
- static std::string getMethodLabelAsString (const EN_ForecastingMethod &)
- static std::string describeLabels ()

32.83.1 Detailed Description

Enumeration of forecasting methods.

Definition at line 15 of file ForecastingMethod.hpp.

32.83.2 Member Enumeration Documentation

32.83.2.1 enum stdair::ForecastingMethod::EN_ForecastingMethod

Enumerator:

Q_FORECASTING

HYBRID_FORECASTING

OLD_QFF

NEW_QFF

BASED_FORECASTING

LAST_VALUE

Definition at line 17 of file ForecastingMethod.hpp.

32.83.3 Constructor & Destructor Documentation

32.83.3.1 stdair::ForecastingMethod::ForecastingMethod (const EN_ForecastingMethod & *iForecastingMethod*)

Constructor.

Definition at line 37 of file ForecastingMethod.cpp.

32.83.3.2 stdair::ForecastingMethod::ForecastingMethod (const char *iMethod*)

Constructor.

Definition at line 42 of file ForecastingMethod.cpp.

References BASED_FORECASTING, describeLabels(), HYBRID_FORECASTING, LAST_VALUE, NEW_QFF, OLD_QFF, and Q_FORECASTING.

32.83.3.3 stdair::ForecastingMethod::ForecastingMethod (const ForecastingMethod & *iForecastingMethod*)

Default copy constructor.

Definition at line 31 of file ForecastingMethod.cpp.

32.83.4 Member Function Documentation

32.83.4.1 const std::string & stdair::ForecastingMethod::getLabel (const EN_ForecastingMethod & *iMethod*) `[static]`

Get the label as a string (e.g., "Q Forecasting", "Hybrid Forecasting", "Old QFF" or "New QFF").

Definition at line 63 of file ForecastingMethod.cpp.

32.83.4.2 char stdair::ForecastingMethod::getMethodLabel (const EN_ForecastingMethod & *iMethod*) `[static]`

Get the label as a single char (e.g., 'Q', 'H', 'O', 'N' or 'B').

Definition at line 68 of file ForecastingMethod.cpp.

32.83.4.3 std::string stdair::ForecastingMethod::getMethodLabelAsString (const EN_ForecastingMethod & *iMethod*) `[static]`

Get the label as a string of a single char (e.g., "Q", "H", "O", "N" or "B").

Definition at line 74 of file ForecastingMethod.cpp.

32.83.4.4 std::string stdair::ForecastingMethod::describeLabels () [static]

List the labels.

Definition at line 81 of file ForecastingMethod.cpp.

References LAST_VALUE.

Referenced by ForecastingMethod().

32.83.4.5 ForecastingMethod::EN_ForecastingMethod stdair::ForecastingMethod::getMethod () const

Get the enumerated value.

Definition at line 93 of file ForecastingMethod.cpp.

Referenced by stdair::AirlineFeature::getForecastingMethod().

32.83.4.6 std::string stdair::ForecastingMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "Q", "H", "O", "N" or "B").

Definition at line 98 of file ForecastingMethod.cpp.

32.83.4.7 const std::string stdair::ForecastingMethod::describe () const [virtual]

Give a description of the structure (e.g., "Q Forecasting", "Hybrid Forecasting, "Old QFF", "New QFF" or "Based Forecasting).

Implements stdair::StructAbstract.

Definition at line 105 of file ForecastingMethod.cpp.

32.83.4.8 bool stdair::ForecastingMethod::operator== (const EN_ForecastingMethod & *iMethod*) const

Comparison operator.

Definition at line 113 of file ForecastingMethod.cpp.

32.83.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.83.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

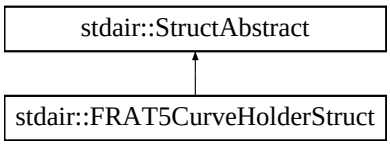
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/ForecastingMethod.hpp
- stdair/basic/ForecastingMethod.cpp

32.84 stdair::FRAT5CurveHolderStruct Struct Reference

`#include <stdair/bom/FRAT5CurveHolderStruct.hpp>`Inheritance diagram for stdair::FRAT5CurveHolderStruct::



Public Member Functions

- const FRAT5Curve_T & getFRAT5Curve (const std::string &) const
- void addCurve (const std::string &, const FRAT5Curve_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- FRAT5CurveHolderStruct ()
- FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct &)
- ~FRAT5CurveHolderStruct ()

32.84.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file FRAT5CurveHolderStruct.hpp.

32.84.2 Constructor & Destructor Documentation

32.84.2.1 stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct ()

Constructor.

Definition at line 14 of file FRAT5CurveHolderStruct.cpp.

32.84.2.2 stdair::FRAT5CurveHolderStruct::FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct & *iHolder*)

Copy constructor.

Definition at line 19 of file FRAT5CurveHolderStruct.cpp.

32.84.2.3 stdair::FRAT5CurveHolderStruct::~FRAT5CurveHolderStruct ()

Destructor.

Definition at line 24 of file FRAT5CurveHolderStruct.cpp.

32.84.3 Member Function Documentation

32.84.3.1 const FRAT5Curve_T & stdair::FRAT5CurveHolderStruct::getFRAT5Curve (const std::string & *iKey*) const

Get the FRAT5 curve corresponding to the given key.

Definition at line 29 of file FRAT5CurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::getFRAT5Curve().

32.84.3.2 void stdair::FRAT5CurveHolderStruct::addCurve (const std::string & *iKey*, const FRAT5Curve_T & *iCurve*)

Add a new curve to the holder.

Definition at line 42 of file FRAT5CurveHolderStruct.cpp.

References STDAIR_LOG_DEBUG.

Referenced by stdair::BomRoot::addFRAT5Curve().

32.84.3.3 void stdair::FRAT5CurveHolderStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 53 of file FRAT5CurveHolderStruct.cpp.

References describe().

32.84.3.4 void stdair::FRAT5CurveHolderStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file FRAT5CurveHolderStruct.cpp.

32.84.3.5 const std::string stdair::FRAT5CurveHolderStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 62 of file FRAT5CurveHolderStruct.cpp.

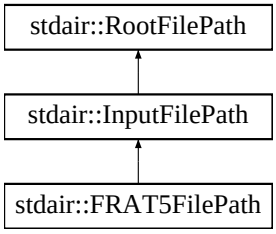
Referenced by toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/FRAT5CurveHolderStruct.hpp
- stdair/bom/FRAT5CurveHolderStruct.cpp

32.85 stdair::FRAT5FilePath Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::FRAT5FilePath::



Public Member Functions

- FRAT5FilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.85.1 Detailed Description

FRAT5 input file.

Definition at line 88 of file stdair_file.hpp.

32.85.2 Constructor & Destructor Documentation

32.85.2.1 stdair::FRAT5FilePath::FRAT5FilePath (const Filename_T & *iFilename*) `[inline, explicit]`

Constructor.

Definition at line 93 of file stdair_file.hpp.

32.85.3 Member Function Documentation

32.85.3.1 const char* stdair::RootFilePath::name () const `[inline, inherited]`

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.85.4 Member Data Documentation

32.85.4.1 const Filename_T stdair::RootFilePath::_filename `[protected, inherited]`

Name of the file.

Definition at line 50 of file stdair_file.hpp.

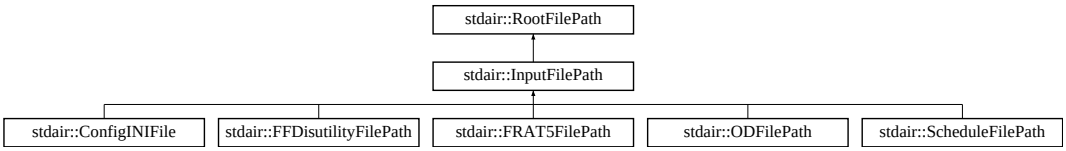
Referenced by stdair::RootFilePath::name().

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

- stdair/stdair_file.hpp

32.86 stdair::InputFilePath Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::InputFilePath::



Public Member Functions

- InputFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.86.1 Detailed Description

Input File.

Definition at line 54 of file stdair_file.hpp.

32.86.2 Constructor & Destructor Documentation

32.86.2.1 stdair::InputFilePath::InputFilePath (const Filename_T & *iFilename*) `[inline]`

Constructor.

Definition at line 57 of file stdair_file.hpp.

32.86.3 Member Function Documentation

32.86.3.1 const char* stdair::RootFilePath::name () const `[inline, inherited]`

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.86.4 Member Data Documentation

32.86.4.1 const Filename_T stdair::RootFilePath::_filename `[protected, inherited]`

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

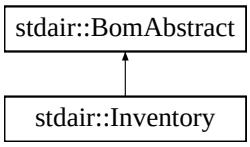
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

32.87 stdair::Inventory Class Reference

Class representing the actual attributes for an airline inventory.

`#include <stdair/bom/Inventory.hpp>`Inheritance diagram for stdair::Inventory::



Public Types

- typedef InventoryKey Key_T

Public Member Functions

- const Key_T & getKey () const
- const AirlineCode_T & getAirlineCode () const
- ForecastingMethod::EN_ForecastingMethod getForecastingMethod () const
- UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod () const
- PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod () const
- OptimisationMethod::EN_OptimisationMethod getOptimisationMethod () const
- PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- FlightDate * getFlightDate (const std::string &iFlightDateKeyStr) const
- FlightDate * getFlightDate (const FlightDateKey &) const
- AirlineFeature * getAirlineFeature () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Inventory (const Key_T &)
- ~Inventory ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- AirlineFeature * _airlineFeature
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.87.1 Detailed Description

Class representing the actual attributes for an airline inventory.

Definition at line 34 of file Inventory.hpp.

32.87.2 Member Typedef Documentation

32.87.2.1 typedef InventoryKey stdair::Inventory::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 45 of file Inventory.hpp.

32.87.3 Constructor & Destructor Documentation

32.87.3.1 stdair::Inventory::Inventory (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 31 of file Inventory.cpp.

32.87.3.2 stdair::Inventory::~Inventory () `[protected]`

Destructor.

Definition at line 38 of file Inventory.cpp.

32.87.4 Member Function Documentation

32.87.4.1 const Key_T& stdair::Inventory::getKey () const `[inline]`

Get the inventory key (airline code).

Definition at line 51 of file Inventory.hpp.

References _key.

32.87.4.2 const AirlineCode_T& stdair::Inventory::getAirlineCode () const `[inline]`

Get the airline code (inventory/primary key).

Definition at line 56 of file Inventory.hpp.

References _key, and stdair::InventoryKey::getAirlineCode().

Referenced by stdair::OnDDate::getAirlineCode(), stdair::FlightDate::getAirlineCode(), stdair::BomJSONExport::jsonExportFlightDateList(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.87.4.3 ForecastingMethod::EN_ForecastingMethod stdair::Inventory::getForecastingMethod () const

Get the forecasting method.

Definition at line 64 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getForecastingMethod().

32.87.4.4 UnconstrainingMethod::EN_UnconstrainingMethod stdair::Inventory::getUnconstrainingMethod () const

Get the unconstraining method.

Definition at line 71 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getUnconstrainingMethod().

32.87.4.5 PreOptimisationMethod::EN_PreOptimisationMethod stdair::Inventory::getPreOptimisationMethod () const

Get the pre-optimisation method.

Definition at line 78 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getPreOptimisationMethod().

32.87.4.6 OptimisationMethod::EN_OptimisationMethod stdair::Inventory::getOptimisationMethod () const

Get the optimisation method.

Definition at line 85 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getOptimisationMethod().

32.87.4.7 PartnershipTechnique::EN_PartnershipTechnique stdair::Inventory::getPartnershipTechnique () const

Get the partnership technique.

Definition at line 92 of file Inventory.cpp.

References _airlineFeature, and stdair::AirlineFeature::getPartnershipTechnique().

32.87.4.8 BomAbstract* const stdair::Inventory::getParent () const [inline]

Get the parent object.

Definition at line 76 of file Inventory.hpp.

References _parent.

32.87.4.9 const HolderMap_T& stdair::Inventory::getHolderMap () const [inline]

Get the map of children.

Definition at line 81 of file Inventory.hpp.

References _holderMap.

32.87.4.10 FlightDate ∗ stdair::Inventory::getFlightDate (const std::string & *iFlightDateKeyStr*) const

Get a pointer on the FlightDate object corresponding to the given key.

Note:

The FlightDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The flight-date key.

Returns:

FlightDate∗ Found FlightDate object. NULL if not found.

Definition at line 50 of file Inventory.cpp.

Referenced by getFlightDate(), stdair::BomRetriever::retrieveFlightDateFromKey(), and stdair::BomRetriever::retrieveFlightDateFromLongKey().

32.87.4.11 FlightDate ∗ stdair::Inventory::getFlightDate (const FlightDateKey & *iFlightDateKey*) const

Get a pointer on the FlightDate object corresponding to the given key.

Note:

The FlightDate object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const FlightDateKey& The flight-date key

Returns:

FlightDate∗ Found FlightDate object. NULL if not found.

Definition at line 58 of file Inventory.cpp.

References getFlightDate(), and stdair::FlightDateKey::toString().

32.87.4.12 AirlineFeature∗ stdair::Inventory::getAirlineFeature () const `[inline]`

Get the airline feature.

Definition at line 112 of file Inventory.hpp.

References _airlineFeature.

32.87.4.13 void stdair::Inventory::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 132 of file Inventory.hpp.

References toString().

32.87.4.14 void stdair::Inventory::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 141 of file Inventory.hpp.

32.87.4.15 std::string stdair::Inventory::toString () const `[virtual]`

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file Inventory.cpp.

References describeKey().

Referenced by toStream().

32.87.4.16 const std::string stdair::Inventory::describeKey () const `[inline]`

Get a string describing the key.

Definition at line 152 of file Inventory.hpp.

References _key, and stdair::InventoryKey::toString().

Referenced by stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), and toString().

32.87.4.17 template<class Archive > void stdair::Inventory::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 160 of file CmdBomSerialiser.cpp.

References _key.

32.87.5 Friends And Related Function Documentation

32.87.5.1 friend class FacBom `[friend]`

Definition at line 35 of file Inventory.hpp.

32.87.5.2 friend class FacCloneBom [friend]

Definition at line 36 of file Inventory.hpp.

32.87.5.3 friend class FacBomManager [friend]

Definition at line 37 of file Inventory.hpp.

32.87.5.4 friend class boost::serialization::access [friend]

Definition at line 38 of file Inventory.hpp.

32.87.6 Member Data Documentation

32.87.6.1 Key_T stdair::Inventory::_key [protected]

Primary key (airline code).

Definition at line 204 of file Inventory.hpp.

Referenced by describeKey(), getAirlineCode(), getKey(), and serialize().

32.87.6.2 BomAbstract* stdair::Inventory::_parent [protected]

Pointer on the parent class (BomRoot).

Definition at line 209 of file Inventory.hpp.

Referenced by getParent().

32.87.6.3 AirlineFeature* stdair::Inventory::_airlineFeature [protected]

Features specific to the airline.

Definition at line 214 of file Inventory.hpp.

Referenced by getAirlineFeature(), getForecastingMethod(), getOptimisationMethod(), getPartnershipTechnique(), getPreOptimisationMethod(), and getUnconstrainingMethod().

32.87.6.4 HolderMap_T stdair::Inventory::_holderMap [protected]

Map holding the children (FlightDate objects).

Definition at line 219 of file Inventory.hpp.

Referenced by getHolderMap().

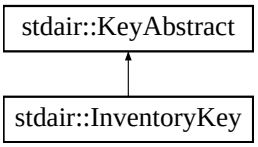
The documentation for this class was generated from the following files:

- stdair/bom/Inventory.hpp
- stdair/bom/Inventory.cpp
- stdair/command/CmdBomSerialiser.cpp

32.88 stdair::InventoryKey Struct Reference

Key of a given inventory, made of the airline code.

`#include <stdair/bom/InventoryKey.hpp>`Inheritance diagram for stdair::InventoryKey::



Public Member Functions

- InventoryKey (const AirlineCode_T &iAirlineCode)
- InventoryKey (const InventoryKey &)
- ~InventoryKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.88.1 Detailed Description

Key of a given inventory, made of the airline code.

Definition at line 26 of file InventoryKey.hpp.

32.88.2 Constructor & Destructor Documentation

32.88.2.1 stdair::InventoryKey::InventoryKey (const AirlineCode_T & *iAirlineCode*)

Constructor.

Definition at line 23 of file InventoryKey.cpp.

32.88.2.2 stdair::InventoryKey::InventoryKey (const InventoryKey & *iKey*)

Copy constructor.

Definition at line 28 of file InventoryKey.cpp.

32.88.2.3 stdair::InventoryKey::~InventoryKey ()

Destructor.

Definition at line 33 of file InventoryKey.cpp.

32.88.3 Member Function Documentation

32.88.3.1 const AirlineCode_T& stdair::InventoryKey::getAirlineCode () const `[inline]`

Get the airline code.

Definition at line 58 of file InventoryKey.hpp.

Referenced by stdair::Inventory::getAirlineCode(), stdair::BomRetriever::retrieveInventoryFromLongKey(), and stdair::BomRetriever::retrievePartnerSegmentDateFromLongKey().

32.88.3.2 void stdair::InventoryKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file InventoryKey.cpp.

References toString().

32.88.3.3 void stdair::InventoryKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file InventoryKey.cpp.

32.88.3.4 const std::string stdair::InventoryKey::toString () const `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file InventoryKey.cpp.

Referenced by stdair::Inventory::describeKey(), stdair::BomRoot::getInventory(), and toStream().

32.88.3.5 template<class Archive > void stdair::InventoryKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 68 of file InventoryKey.cpp.

32.88.4 Friends And Related Function Documentation

32.88.4.1 friend class boost::serialization::access `[friend]`

Definition at line 27 of file InventoryKey.hpp.

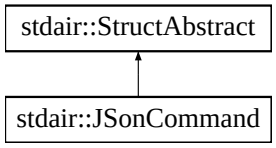
The documentation for this struct was generated from the following files:

- stdair/bom/InventoryKey.hpp
- stdair/bom/InventoryKey.cpp

32.89 stdair::JSonCommand Struct Reference

Enumeration of json commands.

`#include <stdair/basic/JSonCommand.hpp>`Inheritance diagram for stdair::JSonCommand::



Public Types

- enum EN_JSonCommand {
  LIST = 0, FLIGHT_DATE, EVENT_LIST, BREAK_POINT,
  RUN, RESET, STATUS, CONFIG,
  LAST_VALUE }

Public Member Functions

- EN_JSonCommand getCommand () const
- const std::string describe () const

- bool operator== (const EN_JSonCommand &) const
- JSonCommand (const EN_JSonCommand &)
- JSonCommand (const std::string &)
- JSonCommand (const JSonCommand &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static EN_JSonCommand getCommand (const std::string &iCommandStr)
- static std::string getLabel (const EN_JSonCommand &)
- static std::string describeLabels ()

32.89.1 Detailed Description

Enumeration of json commands.

Definition at line 17 of file JSonCommand.hpp.

32.89.2 Member Enumeration Documentation

32.89.2.1 enum stdair::JSonCommand::EN_JSonCommand

Enumerator:

LIST
FLIGHT_DATE
EVENT_LIST
BREAK_POINT
RUN
RESET
STATUS
CONFIG
LAST_VALUE

Definition at line 19 of file JSonCommand.hpp.

32.89.3 Constructor & Destructor Documentation

32.89.3.1 stdair::JSonCommand::JSonCommand (const EN_JSonCommand &)

Main Constructor.

32.89.3.2 stdair::JSonCommand::JSonCommand (const std::string & *iCommandStr*)

Alternative constructor.

Definition at line 71 of file JSonCommand.cpp.

References getCommand().

32.89.3.3 stdair::JSonCommand::JSonCommand (const JSonCommand & *iJSonCommand*)

Default copy constructor.

Definition at line 25 of file JSonCommand.cpp.

32.89.4 Member Function Documentation

32.89.4.1 JSonCommand::EN_JSonCommand stdair::JSonCommand::getCommand (const std::string & *iCommandStr*) `[static]`

Get the command value from parsing a single char (e.g., "list", "flight_date", "event_list", "break_point", "run", "reset, "status" or "config).

Definition at line 31 of file JSonCommand.cpp.

References BREAK_POINT, CONFIG, describeLabels(), EVENT_LIST, FLIGHT_DATE, LAST_-VALUE, LIST, RESET, RUN, and STATUS.

Referenced by stdair::BomJSONImport::jsonImportCommand().

32.89.4.2 std::string stdair::JSonCommand::getLabel (const EN_JSonCommand & *iCommand*) `[static]`

Get a label of a command

Definition at line 66 of file JSonCommand.cpp.

32.89.4.3 std::string stdair::JSonCommand::describeLabels () `[static]`

List the labels.

Definition at line 77 of file JSonCommand.cpp.

References LAST_VALUE.

Referenced by getCommand().

32.89.4.4 JSonCommand::EN_JSonCommand stdair::JSonCommand::getCommand () const

Get the enumerated value.

Definition at line 89 of file JSonCommand.cpp.

Referenced by JSonCommand().

32.89.4.5 const std::string stdair::JSonCommand::describe () const `[virtual]`

Give a description of the structure (e.g., "list", "flight_date", "event_list", "break_point" "run", "reset, "status" or "config).

Implements stdair::StructAbstract.

Definition at line 94 of file JSonCommand.cpp.

32.89.4.6 bool stdair::JSonCommand::operator== (const EN_JSonCommand & *iCommand*) const

Comparison operator.

Definition at line 102 of file JSonCommand.cpp.

32.89.4.7 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.89.4.8 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/JSonCommand.hpp
- stdair/basic/JSonCommand.cpp

32.90 stdair::JSONString Class Reference

JSON-formatted string.

```
#include <stdair/stdair_json.hpp>
```

Public Member Functions

- JSONString (const std::string &iJsonString)
- JSONString ()
- virtual ~JSONString ()
- const std::string & getString () const

Protected Attributes

- std::string _jsonString

32.90.1 Detailed Description

JSON-formatted string.

Definition at line 16 of file stdair_json.hpp.

32.90.2 Constructor & Destructor Documentation

32.90.2.1 stdair::JSONString::JSONString (const std::string & *iJsonString*) `[inline, explicit]`

Main Constructor.

Definition at line 21 of file stdair_json.hpp.

32.90.2.2 stdair::JSONString::JSONString () `[inline, explicit]`

Default constructor.

Definition at line 26 of file stdair_json.hpp.

32.90.2.3 virtual stdair::JSONString::~JSONString () `[inline, virtual]`

Destructor.

Definition at line 31 of file stdair_json.hpp.

32.90.3 Member Function Documentation

32.90.3.1 const std::string& stdair::JSONString::getString () const `[inline]`

Get the string value.

Definition at line 36 of file stdair_json.hpp.

References _jsonString.

Referenced by stdair::BomJSONImport::jsonImportBreakPoints(), stdair::BomJSONImport::jsonImportCommand(), stdair::BomJSONImport::jsonImportConfig(), stdair::BomJSONImport::jsonImportEventType(), stdair::BomJSONImport::jsonImportFlightDate(), stdair::BomJSONImport::jsonImportFlightNumber(), and stdair::BomJSONImport::jsonImportInventoryKey().

32.90.4 Member Data Documentation

32.90.4.1 std::string stdair::JSONString::_jsonString [protected]

Definition at line 44 of file stdair_json.hpp.

Referenced by getString().

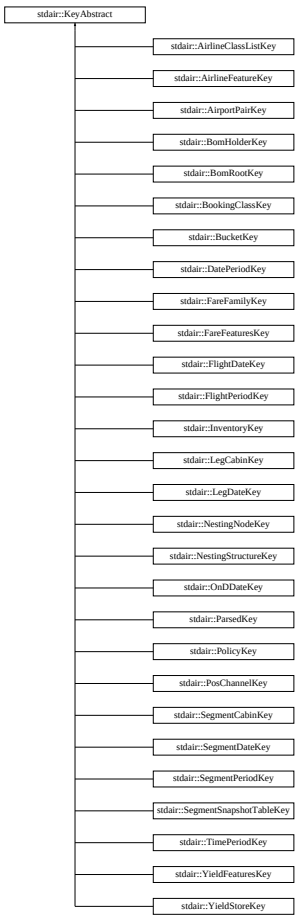
The documentation for this class was generated from the following file:

- stdair/stdair_json.hpp

32.91 stdair::KeyAbstract Struct Reference

Base class for the keys of Business Object Model (BOM) layer.

`#include <stdair/bom/KeyAbstract.hpp>`Inheritance diagram for stdair::KeyAbstract::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const

  Dump a Business Object Key into an output stream.

- virtual void fromStream (std::istream &ioIn)

  Read a Business Object Key from an input stream.

- virtual const std::string toString () const

  Get the serialised version of the Business Object Key.

- virtual ~KeyAbstract ()

  Default destructor.

32.91.1 Detailed Description

Base class for the keys of Business Object Model (BOM) layer. Note that that key allows to differentiate two objects at the same level only. For instance, the segment-date key allows to differentiate two segment-dates under a given flight-date, but does not allow to differentiate two segment-dates in general.

Definition at line 27 of file KeyAbstract.hpp.

32.91.2 Constructor & Destructor Documentation

32.91.2.1 virtual stdair::KeyAbstract::~KeyAbstract () `[inline, virtual]`

Default destructor.

Definition at line 61 of file KeyAbstract.hpp.

32.91.3 Member Function Documentation

32.91.3.1 virtual void stdair::KeyAbstract::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object Key into an output stream.

Parameters:

↔ ***ostream&*** the output stream.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

Definition at line 36 of file KeyAbstract.hpp.

32.91.3.2 virtual void stdair::KeyAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object Key from an input stream.

Parameters:

↔ ***istream&*** the input stream.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

Definition at line 43 of file KeyAbstract.hpp.

Referenced by operator>>().

32.91.3.3 virtual const std::string stdair::KeyAbstract::toString () const [inline, virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Parameters:

→ ***const*** std::string The serialised version of the Business Object Key.

Reimplemented in stdair::AirlineClassListKey, stdair::AirlineFeatureKey, stdair::AirportPairKey, stdair::BomHolderKey, stdair::BomRootKey, stdair::BookingClassKey, stdair::BucketKey, stdair::DatePeriodKey, stdair::FareFamilyKey, stdair::FareFeaturesKey, stdair::FlightDateKey, stdair::FlightPeriodKey, stdair::InventoryKey, stdair::LegCabinKey, stdair::LegDateKey, stdair::NestingNodeKey, stdair::NestingStructureKey, stdair::OnDDateKey, stdair::ParsedKey, stdair::PolicyKey, stdair::PosChannelKey, stdair::SegmentCabinKey, stdair::SegmentDateKey, stdair::SegmentPeriodKey, stdair::SegmentSnapshotTableKey, stdair::TimePeriodKey, stdair::YieldFeaturesKey, and stdair::YieldStoreKey.

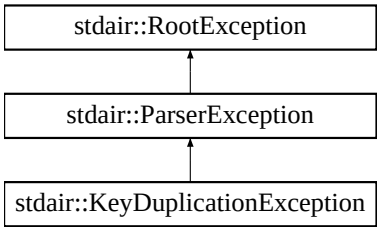
Definition at line 56 of file KeyAbstract.hpp.

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

- stdair/bom/KeyAbstract.hpp

32.92 stdair::KeyDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::KeyDuplicationException::



Public Member Functions

- KeyDuplicationException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.92.1 Detailed Description

Key duplication.

Definition at line 149 of file stdair_exceptions.hpp.

32.92.2 Constructor & Destructor Documentation

32.92.2.1 stdair::KeyDuplicationException::KeyDuplicationException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 152 of file stdair_exceptions.hpp.

32.92.3 Member Function Documentation

32.92.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.92.4 Member Data Documentation

32.92.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

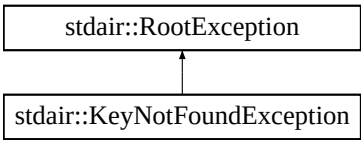
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.93 stdair::KeyNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::KeyNotFoundException::



Public Member Functions

- KeyNotFoundException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.93.1 Detailed Description

Not found key.

Definition at line 126 of file stdair_exceptions.hpp.

32.93.2 Constructor & Destructor Documentation

32.93.2.1 stdair::KeyNotFoundException::KeyNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 129 of file stdair_exceptions.hpp.

32.93.3 Member Function Documentation

32.93.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.93.4 Member Data Documentation

32.93.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

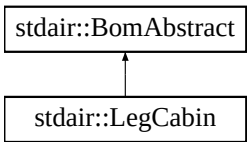
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

32.94 stdair::LegCabin Class Reference

Class representing the actual attributes for an airline leg-cabin.

`#include <stdair/bom/LegCabin.hpp>`Inheritance diagram for stdair::LegCabin::



Public Types

- typedef LegCabinKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const CabinCode_T & getCabinCode () const
- const MapKey_T getFullerKey () const
- const HolderMap_T & getHolderMap () const
- const CabinCapacity_T & getOfferedCapacity () const
- const CabinCapacity_T & getPhysicalCapacity () const
- const NbOfSeats_T & getSoldSeat () const
- const CommittedSpace_T & getCommittedSpace () const
- const Availability_T & getAvailabilityPool () const
- const Availability_T & getAvailability () const
- const BidPrice_T & getCurrentBidPrice () const
- const BidPrice_T & getPreviousBidPrice () const
- const BidPriceVector_T & getBidPriceVector () const
- const CapacityAdjustment_T & getRegradeAdjustment () const
- const AuthorizationLevel_T & getAuthorizationLevel () const
- const UPR_T & getUPR () const
- const Availability_T & getNetAvailability () const
- const Availability_T & getGrossAvailability () const

- const OverbookingRate_T & getAvgCancellationPercentage () const
- const NbOfSeats_T & getETB () const
- const NbOfSeats_T & getStaffNbOfSeats () const
- const NbOfSeats_T & getWLNbOfSeats () const
- const NbOfSeats_T & getGroupNbOfSeats () const
- VirtualClassList_T & getVirtualClassList ()
- BidPriceVector_T & getBidPriceVector ()
- const YieldLevelDemandMap_T & getYieldLevelDemandMap ()
- void setCapacities (const CabinCapacity_T &iCapacity)
- void setSoldSeat (const NbOfSeats_T &iSoldSeat)
- void setCommittedSpace (const CommittedSpace_T &iCommittedSpace)
- void setAvailabilityPool (const Availability_T &iAvailabilityPool)
- void setAvailability (const Availability_T &iAvailability)
- void setCurrentBidPrice (const BidPrice_T &iBidPrice)
- void setPreviousBidPrice (const BidPrice_T &iBidPrice)
- void updatePreviousBidPrice ()
- void setRegradeAdjustment (const CapacityAdjustment_T &iRegradeAdjustment)
- void setAuthorizationLevel (const AuthorizationLevel_T &iAU)
- void setUPR (const UPR_T &iUPR)
- void setNetAvailability (const Availability_T &iNAV)
- void setGrossAvailability (const Availability_T &iGAV)
- void setAvgCancellationPercentage (const OverbookingRate_T &iACP)
- void setETB (const NbOfSeats_T &iETB)
- void setStaffNbOfSeats (const NbOfSeats_T &iStaffSeats)
- void setWLNbOfSeats (const NbOfSeats_T &iWLSeats)
- void setGroupNbOfSeats (const NbOfSeats_T &iGroupSeats)
- void updateCurrentBidPrice ()
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string displayVirtualClassList () const
- void updateFromReservation (const NbOfBookings_T &)
- void addVirtualClass (const VirtualClassStruct &iVC)
- void emptyVirtualClassList ()
- void emptyBidPriceVector ()
- void addDemandInformation (const YieldValue_T &, const MeanValue_T &, const StdDevValue_T &)
- void emptyYieldLevelDemandMap ()

Public Attributes

- CapacityAdjustment_T _dcsRegrade
- AuthorizationLevel_T _au
- UPR_T _upr
- Availability_T _nav
- Availability_T _gav
- OverbookingRate_T _acp
- NbOfSeats_T _etb
- NbOfSeats_T _staffNbOfBookings
- NbOfSeats_T _wlNbOfBookings
- NbOfSeats_T _groupNbOfBookings

Protected Member Functions

- LegCabin (const Key_T &)
- ~LegCabin ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- CabinCapacity_T _offeredCapacity
- CabinCapacity_T _physicalCapacity
- NbOfSeats_T _soldSeat
- CommittedSpace_T _committedSpace
- Availability_T _availabilityPool
- Availability_T _availability
- BidPrice_T _currentBidPrice
- BidPrice_T _previousBidPrice
- BidPriceVector_T _bidPriceVector
- VirtualClassList_T _virtualClassList
- YieldLevelDemandMap_T _yieldLevelDemandMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.94.1 Detailed Description

Class representing the actual attributes for an airline leg-cabin.

Definition at line 25 of file LegCabin.hpp.

32.94.2 Member Typedef Documentation

32.94.2.1 typedef LegCabinKey stdair::LegCabin::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 35 of file LegCabin.hpp.

32.94.3 Constructor & Destructor Documentation

32.94.3.1 stdair::LegCabin::LegCabin (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 46 of file LegCabin.cpp.

32.94.3.2 stdair::LegCabin::~LegCabin () [protected]

Destructor.

Definition at line 69 of file LegCabin.cpp.

32.94.4 Member Function Documentation

32.94.4.1 const Key_T& stdair::LegCabin::getKey () const [inline]

Get the leg-cabin key (cabin code).

Definition at line 42 of file LegCabin.hpp.

References _key.

32.94.4.2 BomAbstract* const stdair::LegCabin::getParent () const [inline]

Get the parent object.

Definition at line 49 of file LegCabin.hpp.

References _parent.

32.94.4.3 const CabinCode_T& stdair::LegCabin::getCabinCode () const [inline]

Get the cabin code (from key).

Definition at line 56 of file LegCabin.hpp.

References _key, and stdair::LegCabinKey::getCabinCode().

Referenced by getFullerKey().

32.94.4.4 const MapKey_T stdair::LegCabin::getFullerKey () const

Get the (leg-date, leg-cabin) key (board point and cabin code).

Note:

That method assumes that the parent object derives from the SegmentDate class, as it needs to have access to the describeKey() method.

Definition at line 80 of file LegCabin.cpp.

References stdair::DEFAULT_KEY_FLD_DELIMITER, stdair::LegDate::describeKey(), and getCabinCode().

32.94.4.5 const HolderMap_T& stdair::LegCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 72 of file LegCabin.hpp.

References _holderMap.

32.94.4.6 const CabinCapacity_T& stdair::LegCabin::getOfferedCapacity () const [inline]

Get the cabin offered capacity.

Definition at line 77 of file LegCabin.hpp.

References _offeredCapacity.

32.94.4.7 const CabinCapacity_T& stdair::LegCabin::getPhysicalCapacity () const [inline]

Get the cabin physical capacity.

Definition at line 82 of file LegCabin.hpp.

References _physicalCapacity.

32.94.4.8 const NbOfSeats_T& stdair::LegCabin::getSoldSeat () const [inline]

Get the number of sold seat.

Definition at line 87 of file LegCabin.hpp.

References _soldSeat.

32.94.4.9 const CommittedSpace_T& stdair::LegCabin::getCommittedSpace () const [inline]

Get the value of committed space.

Definition at line 92 of file LegCabin.hpp.

References _committedSpace.

32.94.4.10 const Availability_T& stdair::LegCabin::getAvailabilityPool () const [inline]

Get the value of the availability pool.

Definition at line 97 of file LegCabin.hpp.

References _availabilityPool.

32.94.4.11 const Availability_T& stdair::LegCabin::getAvailability () const [inline]

Get the value of the availability.

Definition at line 102 of file LegCabin.hpp.

References _availability.

32.94.4.12 const BidPrice_T& stdair::LegCabin::getCurrentBidPrice () const [inline]

Get the current Bid-Price.

Definition at line 107 of file LegCabin.hpp.

References _currentBidPrice.

32.94.4.13 const BidPrice_T& stdair::LegCabin::getPreviousBidPrice () const [inline]

Get the previous Bid-Price.

Definition at line 112 of file LegCabin.hpp.

References _previousBidPrice.

32.94.4.14 const BidPriceVector_T& stdair::LegCabin::getBidPriceVector () const [inline]

Get the Bid-Price Vector.

Definition at line 117 of file LegCabin.hpp.

References _bidPriceVector.

32.94.4.15 const CapacityAdjustment_T& stdair::LegCabin::getRegradeAdjustment () const [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 122 of file LegCabin.hpp.

References _dcsRegrade.

32.94.4.16 const AuthorizationLevel_T& stdair::LegCabin::getAuthorizationLevel () const [inline]

Authorisation Level (AU).

Definition at line 127 of file LegCabin.hpp.

References _au.

32.94.4.17 const UPR_T& stdair::LegCabin::getUPR () const [inline]

Unsold Protection (UPR).

Definition at line 132 of file LegCabin.hpp.

References _upr.

32.94.4.18 const Availability_T& stdair::LegCabin::getNetAvailability () const [inline]

Net Availability (NAV).

Definition at line 137 of file LegCabin.hpp.

References _nav.

32.94.4.19 const Availability_T& stdair::LegCabin::getGrossAvailability () const [inline]

Gross Availability (GAV).

Definition at line 142 of file LegCabin.hpp.

References _gav.

32.94.4.20 const OverbookingRate_T& stdair::LegCabin::getAvgCancellationPercentage () const [inline]

Average Cancellation Percentage (ACP).

Definition at line 147 of file LegCabin.hpp.

References _acp.

32.94.4.21 const NbOfSeats_T& stdair::LegCabin::getETB () const [inline]

Expected to Board (ETB).

Definition at line 152 of file LegCabin.hpp.

References _etb.

32.94.4.22 const NbOfSeats_T& stdair::LegCabin::getStaffNbOfSeats () const [inline]

Number of staff bookings.

Definition at line 157 of file LegCabin.hpp.

References _staffNbOfBookings.

32.94.4.23 const NbOfSeats_T& stdair::LegCabin::getWLNbOfSeats () const [inline]

Number of wait-listed bookings.

Definition at line 162 of file LegCabin.hpp.

References _wlNbOfBookings.

32.94.4.24 const NbOfSeats_T& stdair::LegCabin::getGroupNbOfSeats () const [inline]

Number of group bookings.

Definition at line 167 of file LegCabin.hpp.

References _groupNbOfBookings.

32.94.4.25 VirtualClassList_T& stdair::LegCabin::getVirtualClassList () [inline]

The virtual class list.

Definition at line 172 of file LegCabin.hpp.

References _virtualClassList.

32.94.4.26 BidPriceVector_T& stdair::LegCabin::getBidPriceVector () [inline]

Reset the bid price vector and return it.

Definition at line 177 of file LegCabin.hpp.

References _bidPriceVector.

32.94.4.27 const YieldLevelDemandMap_T& stdair::LegCabin::getYieldLevelDemandMap () [inline]

Get the yield-demand map.

Definition at line 183 of file LegCabin.hpp.

References _yieldLevelDemandMap.

32.94.4.28 void stdair::LegCabin::setCapacities (const CabinCapacity_T & *iCapacity*)

Set the offered and physical capacities.

Definition at line 73 of file LegCabin.cpp.

References _committedSpace, _offeredCapacity, _physicalCapacity, and setAvailabilityPool().

32.94.4.29 void stdair::LegCabin::setSoldSeat (const NbOfSeats_T & *iSoldSeat*) [inline]

Set the number of sold seat.

Definition at line 194 of file LegCabin.hpp.

References _soldSeat.

32.94.4.30 void stdair::LegCabin::setCommittedSpace (const CommittedSpace_T & *iCommittedSpace*) [inline]

Set the value of committed space.

Definition at line 199 of file LegCabin.hpp.

References _committedSpace.

32.94.4.31 void stdair::LegCabin::setAvailabilityPool (const Availability_T & *iAvailabilityPool*) [inline]

Set the value of availability pool.

Definition at line 204 of file LegCabin.hpp.

References _availabilityPool.

Referenced by setCapacities().

32.94.4.32 void stdair::LegCabin::setAvailability (const Availability_T & *iAvailability*) [inline]

Set the value of availability.

Definition at line 209 of file LegCabin.hpp.

References _availability.

32.94.4.33 void stdair::LegCabin::setCurrentBidPrice (const BidPrice_T & *iBidPrice*) [inline]

Set the current Bid-Price.

Definition at line 214 of file LegCabin.hpp.

References _currentBidPrice.

32.94.4.34 void stdair::LegCabin::setPreviousBidPrice (const BidPrice_T & *iBidPrice*) [inline]

Set the previous Bid-Price.

Definition at line 219 of file LegCabin.hpp.

References _previousBidPrice.

32.94.4.35 void stdair::LegCabin::updatePreviousBidPrice () [inline]

Update the previous bid price value with the current one.

Definition at line 224 of file LegCabin.hpp.

References _currentBidPrice, and _previousBidPrice.

32.94.4.36 void stdair::LegCabin::setRegradeAdjustment (const CapacityAdjustment_T & *iRegradeAdjustment*) [inline]

Get the capacity adjustment due to check-in (DCS) regrade.

Definition at line 229 of file LegCabin.hpp.

References _dcsRegrade.

32.94.4.37 void stdair::LegCabin::setAuthorizationLevel (const AuthorizationLevel_T & *iAU*) [inline]

Set the Authorisation Level (AU).

Definition at line 234 of file LegCabin.hpp.

References _au.

32.94.4.38 void stdair::LegCabin::setUPR (const UPR_T & *iUPR*) [inline]

Set the Unsold Protection (UPR).

Definition at line 239 of file LegCabin.hpp.

References _upr.

32.94.4.39 void stdair::LegCabin::setNetAvailability (const Availability_T & *iNAV*) [inline]

Set the Net Availability (NAV).

Definition at line 244 of file LegCabin.hpp.

References _nav.

32.94.4.40 void stdair::LegCabin::setGrossAvailability (const Availability_T & *iGAV*) [inline]

Set the Gross Availability (GAV).

Definition at line 249 of file LegCabin.hpp.

References _gav.

32.94.4.41 void stdair::LegCabin::setAvgCancellationPercentage (const OverbookingRate_T & *iACP*) [inline]

Set the Average Cancellation Percentage (ACP).

Definition at line 254 of file LegCabin.hpp.

References _acp.

32.94.4.42 void stdair::LegCabin::setETB (const NbOfSeats_T & *iETB*) [inline]

Set the Expected to Board (ETB).

Definition at line 259 of file LegCabin.hpp.

References _etb.

32.94.4.43 void stdair::LegCabin::setStaffNbOfSeats (const NbOfSeats_T & *iStaffSeats*) [inline]

Set the Number of staff sold seats.

Definition at line 264 of file LegCabin.hpp.

References _staffNbOfBookings.

32.94.4.44 void stdair::LegCabin::setWLNbOfSeats (const NbOfSeats_T & *iWLSeats*) [inline]

Set the Number of wait-listed sold seats.

Definition at line 269 of file LegCabin.hpp.

References _wlNbOfBookings.

32.94.4.45 void stdair::LegCabin::setGroupNbOfSeats (const NbOfSeats_T & *iGroupSeats*) [inline]

Set the Number of group sold seats.

Definition at line 274 of file LegCabin.hpp.

References _groupNbOfBookings.

32.94.4.46 void stdair::LegCabin::updateCurrentBidPrice ()

Update the bid price (from bid price vector if not empty).

Definition at line 120 of file LegCabin.cpp.

References _availabilityPool, _bidPriceVector, and _currentBidPrice.

32.94.4.47 void stdair::LegCabin::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 288 of file LegCabin.hpp.

References toString().

32.94.4.48 void stdair::LegCabin::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 296 of file LegCabin.hpp.

32.94.4.49 std::string stdair::LegCabin::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 89 of file LegCabin.cpp.

References describeKey().

Referenced by toStream().

32.94.4.50 const std::string stdair::LegCabin::describeKey () const [inline]

Get a string describing the key.

Definition at line 307 of file LegCabin.hpp.

References _key, and stdair::LegCabinKey::toString().

Referenced by toString().

32.94.4.51 const std::string stdair::LegCabin::displayVirtualClassList () const

Display the virtual class list content.

Definition at line 96 of file LegCabin.cpp.

References _virtualClassList, stdair::VirtualClassStruct::getCumulatedBookingLimit(), stdair::VirtualClassStruct::getCumulatedProtection(), and stdair::VirtualClassStruct::getYield().

32.94.4.52 void stdair::LegCabin::updateFromReservation (const NbOfBookings_T & *iNbOfBookings*)

Register a sale.

Definition at line 114 of file LegCabin.cpp.

References _availabilityPool, _committedSpace, and _offeredCapacity.

32.94.4.53 void stdair::LegCabin::addVirtualClass (const VirtualClassStruct & *iVC*) [inline]

Add a virtual class to the list.

Definition at line 327 of file LegCabin.hpp.

References _virtualClassList.

32.94.4.54 void stdair::LegCabin::emptyVirtualClassList () [inline]

Empty the virtual class list.

Definition at line 334 of file LegCabin.hpp.

References _virtualClassList.

32.94.4.55 void stdair::LegCabin::emptyBidPriceVector () [inline]

Empty the bid price vector.

Definition at line 341 of file LegCabin.hpp.

References _bidPriceVector.

32.94.4.56 void stdair::LegCabin::addDemandInformation (const YieldValue_T & *iYield*, const MeanValue_T & *iMeanValue*, const StdDevValue_T & *iStdDevValue*)

Add demand information.

Definition at line 133 of file LegCabin.cpp.

References _yieldLevelDemandMap.

32.94.4.57 void stdair::LegCabin::emptyYieldLevelDemandMap () [inline]

Reset the (yield level,demand) map.

Definition at line 354 of file LegCabin.hpp.

References _yieldLevelDemandMap.

32.94.5 Friends And Related Function Documentation

32.94.5.1 friend class FacBom [friend]

Definition at line 26 of file LegCabin.hpp.

32.94.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file LegCabin.hpp.

32.94.5.3 friend class FacBomManager [friend]

Definition at line 28 of file LegCabin.hpp.

32.94.6 Member Data Documentation

32.94.6.1 Key_T stdair::LegCabin::_key [protected]

Primary key (cabin code).

Definition at line 387 of file LegCabin.hpp.

Referenced by describeKey(), getCabinCode(), and getKey().

32.94.6.2 BomAbstract* stdair::LegCabin::_parent [protected]

Pointer on the parent class (LegDate).

Definition at line 392 of file LegCabin.hpp.

Referenced by getParent().

32.94.6.3 HolderMap_T stdair::LegCabin::_holderMap [protected]

Map holding the children (Bucket objects).

Definition at line 397 of file LegCabin.hpp.

Referenced by getHolderMap().

32.94.6.4 CabinCapacity_T stdair::LegCabin::_offeredCapacity [protected]

Saleable capacity of the cabin.

Definition at line 400 of file LegCabin.hpp.

Referenced by getOfferedCapacity(), setCapacities(), and updateFromReservation().

32.94.6.5 CabinCapacity_T stdair::LegCabin::_physicalCapacity [protected]

Physical capacity of the cabin.

Definition at line 403 of file LegCabin.hpp.

Referenced by getPhysicalCapacity(), and setCapacities().

32.94.6.6 NbOfSeats_T stdair::LegCabin::_soldSeat [protected]

Aggregated number of sold seats.

Definition at line 406 of file LegCabin.hpp.

Referenced by getSoldSeat(), and setSoldSeat().

32.94.6.7 CommittedSpace_T stdair::LegCabin::_committedSpace [protected]

Definition at line 409 of file LegCabin.hpp.

Referenced by getCommittedSpace(), setCapacities(), setCommittedSpace(), and updateFromReservation().

32.94.6.8 Availability_T stdair::LegCabin::_availabilityPool [protected]

Availability pool.

Definition at line 412 of file LegCabin.hpp.

Referenced by getAvailabilityPool(), setAvailabilityPool(), updateCurrentBidPrice(), and updateFromReservation().

32.94.6.9 Availability_T stdair::LegCabin::_availability [protected]

Availability.

Definition at line 415 of file LegCabin.hpp.

Referenced by getAvailability(), and setAvailability().

32.94.6.10 BidPrice_T stdair::LegCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 418 of file LegCabin.hpp.

Referenced by getCurrentBidPrice(), setCurrentBidPrice(), updateCurrentBidPrice(), and updatePreviousBidPrice().

32.94.6.11 BidPrice_T stdair::LegCabin::_previousBidPrice [protected]

Previous Bid-Price (BP).

Definition at line 421 of file LegCabin.hpp.

Referenced by getPreviousBidPrice(), setPreviousBidPrice(), and updatePreviousBidPrice().

32.94.6.12 BidPriceVector_T stdair::LegCabin::_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 424 of file LegCabin.hpp.

Referenced by emptyBidPriceVector(), getBidPriceVector(), and updateCurrentBidPrice().

32.94.6.13 VirtualClassList_T stdair::LegCabin::_virtualClassList [protected]

List of virtual classes (for revenue management optimisation).

Definition at line 427 of file LegCabin.hpp.

Referenced by addVirtualClass(), displayVirtualClassList(), emptyVirtualClassList(), and getVirtualClassList().

32.94.6.14 YieldLevelDemandMap_T stdair::LegCabin::_yieldLevelDemandMap [protected]

Map holding the demand information indexed by yield.

Definition at line 430 of file LegCabin.hpp.

Referenced by addDemandInformation(), emptyYieldLevelDemandMap(), and getYieldLevelDemandMap().

32.94.6.15 CapacityAdjustment_T stdair::LegCabin::_dcsRegrade

Capacity adjustment of the cabin, due to check-in (DCS) regrade.

Definition at line 435 of file LegCabin.hpp.

Referenced by getRegradeAdjustment(), and setRegradeAdjustment().

32.94.6.16 AuthorizationLevel_T stdair::LegCabin::_au

Authorisation Level (AU).

Definition at line 438 of file LegCabin.hpp.

Referenced by getAuthorizationLevel(), and setAuthorizationLevel().

32.94.6.17 UPR_T stdair::LegCabin::_upr

Unsold Protection (UPR).

Definition at line 441 of file LegCabin.hpp.

Referenced by getUPR(), and setUPR().

32.94.6.18 Availability_T stdair::LegCabin::_nav

Net Availability (NAV).

Definition at line 444 of file LegCabin.hpp.

Referenced by getNetAvailability(), and setNetAvailability().

32.94.6.19 Availability_T stdair::LegCabin::_gav

Gross Availability (GAV).

Definition at line 447 of file LegCabin.hpp.

Referenced by getGrossAvailability(), and setGrossAvailability().

32.94.6.20 OverbookingRate_T stdair::LegCabin::_acp

Average Cancellation Percentage (ACP).

Definition at line 450 of file LegCabin.hpp.

Referenced by getAvgCancellationPercentage(), and setAvgCancellationPercentage().

32.94.6.21 NbOfSeats_T stdair::LegCabin::_etb

Expected to Board (ETB).

Definition at line 453 of file LegCabin.hpp.

Referenced by getETB(), and setETB().

32.94.6.22 NbOfSeats_T stdair::LegCabin::_staffNbOfBookings

Number of staff bookings.

Definition at line 456 of file LegCabin.hpp.

Referenced by getStaffNbOfSeats(), and setStaffNbOfSeats().

32.94.6.23 NbOfSeats_T stdair::LegCabin::_wlNbOfBookings

Number of wait-listed bookings.

Definition at line 459 of file LegCabin.hpp.

Referenced by getWLNbOfSeats(), and setWLNbOfSeats().

32.94.6.24 NbOfSeats_T stdair::LegCabin::_groupNbOfBookings

Number of group bookings.

Definition at line 462 of file LegCabin.hpp.

Referenced by getGroupNbOfSeats(), and setGroupNbOfSeats().

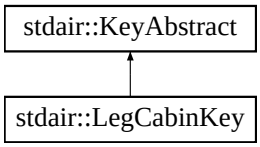
The documentation for this class was generated from the following files:

- stdair/bom/LegCabin.hpp
- stdair/bom/LegCabin.cpp

32.95 stdair::LegCabinKey Struct Reference

Key of a given leg-cabin, made of a cabin code (only).

`#include <stdair/bom/LegCabinKey.hpp>`Inheritance diagram for stdair::LegCabinKey::



Public Member Functions

- LegCabinKey (const CabinCode_T &iCabinCode)
- LegCabinKey (const LegCabinKey &)
- ~LegCabinKey ()
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.95.1 Detailed Description

Key of a given leg-cabin, made of a cabin code (only).

Definition at line 26 of file LegCabinKey.hpp.

32.95.2 Constructor & Destructor Documentation

32.95.2.1 stdair::LegCabinKey::LegCabinKey (const CabinCode_T & *iCabinCode*)

Constructor.

Definition at line 23 of file LegCabinKey.cpp.

32.95.2.2 stdair::LegCabinKey::LegCabinKey (const LegCabinKey & *iKey*)

Copy constructor.

Definition at line 28 of file LegCabinKey.cpp.

32.95.2.3 stdair::LegCabinKey::~LegCabinKey ()

Destructor.

Definition at line 33 of file LegCabinKey.cpp.

32.95.3 Member Function Documentation

32.95.3.1 const CabinCode_T& stdair::LegCabinKey::getCabinCode () const [inline]

Get the cabin code.

Definition at line 56 of file LegCabinKey.hpp.

Referenced by stdair::LegCabin::getCabinCode().

32.95.3.2 void stdair::LegCabinKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file LegCabinKey.cpp.

References toString().

32.95.3.3 void stdair::LegCabinKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file LegCabinKey.cpp.

32.95.3.4 const std::string stdair::LegCabinKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file LegCabinKey.cpp.

Referenced by stdair::LegCabin::describeKey(), stdair::LegDate::getLegCabin(), and toStream().

32.95.3.5 template<class Archive > void stdair::LegCabinKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 68 of file LegCabinKey.cpp.

32.95.4 Friends And Related Function Documentation

32.95.4.1 friend class boost::serialization::access [friend]

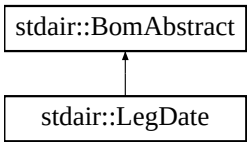
Definition at line 27 of file LegCabinKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/LegCabinKey.hpp
- stdair/bom/LegCabinKey.cpp

32.96 stdair::LegDate Class Reference

`#include <stdair/bom/LegDate.hpp>`Inheritance diagram for stdair::LegDate::



Public Types

- typedef LegDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirlineCode_T & getAirlineCode () const
- const HolderMap_T & getHolderMap () const
- LegCabin * getLegCabin (const std::string &iLegCabinKeyStr) const
- LegCabin * getLegCabin (const LegCabinKey &) const
- const AirportCode_T & getOffPoint () const
- const Date_T & getBoardingDate () const
- const Duration_T & getBoardingTime () const
- const Date_T & getOffDate () const
- const Duration_T & getOffTime () const
- const Duration_T & getElapsedTime () const
- const Distance_T & getDistance () const

- const CabinCapacity_T & getCapacity () const
- const DateOffset_T getDateOffset () const
- const Duration_T getTimeOffset () const
- void setOffPoint (const AirportCode_T &iOffPoint)
- void setBoardingDate (const Date_T &iBoardingDate)
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffDate (const Date_T &iOffDate)
- void setOffTime (const Duration_T &iOffTime)
- void setElapsedTime (const Duration_T &)
- void setOperatingAirlineCode (const AirlineCode_T &iAirlineCode)
- void setOperatingFlightNumber (const FlightNumber_T &iFlightNumber)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string describeRoutingKey () const

Protected Member Functions

- LegDate (const Key_T &)
- virtual ~LegDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- AirportCode_T _offPoint
- Date_T _boardingDate
- Duration_T _boardingTime
- Date_T _offDate
- Duration_T _offTime
- Duration_T _elapsedTime
- Distance_T _distance
- CabinCapacity_T _capacity
- AirlineCode_T _operatingAirlineCode
- FlightNumber_T _operatingFlightNumber

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.96.1 Detailed Description

Class representing the actual attributes for an airline leg-date.

Definition at line 25 of file LegDate.hpp.

32.96.2 Member Typedef Documentation

32.96.2.1 typedef LegDateKey stdair::LegDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 33 of file LegDate.hpp.

32.96.3 Constructor & Destructor Documentation

32.96.3.1 stdair::LegDate::LegDate (const Key_T & *iKey*) [protected]

Constructor.

Definition at line 38 of file LegDate.cpp.

32.96.3.2 stdair::LegDate::∼LegDate () [protected, virtual]

Destructor.

Definition at line 44 of file LegDate.cpp.

32.96.4 Member Function Documentation

32.96.4.1 const Key_T& stdair::LegDate::getKey () const [inline]

Get the leg-date key.

Definition at line 39 of file LegDate.hpp.

References _key.

32.96.4.2 BomAbstract∗ const stdair::LegDate::getParent () const [inline]

Get the parent object.

Definition at line 44 of file LegDate.hpp.

References _parent.

Referenced by describeRoutingKey(), and getAirlineCode().

32.96.4.3 const AirportCode_T& stdair::LegDate::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 49 of file LegDate.hpp.

References _key, and stdair::LegDateKey::getBoardingPoint().

32.96.4.4 const AirlineCode_T & stdair::LegDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 48 of file LegDate.cpp.

References stdair::FlightDate::getAirlineCode(), and getParent().

32.96.4.5 const HolderMap_T& stdair::LegDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 65 of file LegDate.hpp.

References _holderMap.

32.96.4.6 LegCabin * stdair::LegDate::getLegCabin (const std::string & *iLegCabinKeyStr*) const

Get a pointer on the LegCabin object corresponding to the given key.

Note:

The LegCabin object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const std::string& The leg-cabin key.

Returns:

LegCabin* Found LegCabin object. NULL if not found.

Definition at line 76 of file LegDate.cpp.

Referenced by getLegCabin(), and stdair::BomRetriever::retrieveDummyLegCabin().

32.96.4.7 LegCabin * stdair::LegDate::getLegCabin (const LegCabinKey & *iLegCabinKey*) const

Get a pointer on the LegCabin object corresponding to the given key.

Note:

The LegCabin object can be inherited from, if needed. In that case, a dynamic_cast<> may be needed.

Parameters:

const LegCabinKey& The leg-cabin key

Returns:

LegCabin* Found LegCabin object. NULL if not found.

Definition at line 83 of file LegDate.cpp.

References getLegCabin(), and stdair::LegCabinKey::toString().

32.96.4.8 const AirportCode_T& stdair::LegDate::getOffPoint () const [inline]

Get the off point.

Definition at line 94 of file LegDate.hpp.

References _offPoint.

32.96.4.9 const Date_T& stdair::LegDate::getBoardingDate () const [inline]

Get the boarding date.

Definition at line 99 of file LegDate.hpp.

References _boardingDate.

32.96.4.10 const Duration_T& stdair::LegDate::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 104 of file LegDate.hpp.

References _boardingTime.

32.96.4.11 const Date_T& stdair::LegDate::getOffDate () const [inline]

Get the off date.

Definition at line 109 of file LegDate.hpp.

References _offDate.

32.96.4.12 const Duration_T& stdair::LegDate::getOffTime () const [inline]

Get the off time.

Definition at line 114 of file LegDate.hpp.

References _offTime.

32.96.4.13 const Duration_T& stdair::LegDate::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 119 of file LegDate.hpp.

References _elapsedTime.

32.96.4.14 const Distance_T& stdair::LegDate::getDistance () const [inline]

Get the distance.

Definition at line 124 of file LegDate.hpp.

References _distance.

32.96.4.15 const CabinCapacity_T& stdair::LegDate::getCapacity () const [inline]

Get the leg capacity.

Definition at line 129 of file LegDate.hpp.

References _capacity.

32.96.4.16 const DateOffset_T stdair::LegDate::getDateOffset () const [inline]

Get the date offset (off date - boarding date).

Definition at line 134 of file LegDate.hpp.

References _boardingDate, and _offDate.

Referenced by getTimeOffset().

32.96.4.17 const Duration_T stdair::LegDate::getTimeOffset () const

Get the time off set between boarding and off points.

It is defined as being: TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) $*$ 24

- ElapsedTime.

Definition at line 88 of file LegDate.cpp.

References _boardingTime, _elapsedTime, _offTime, and getDateOffset().

32.96.4.18 void stdair::LegDate::setOffPoint (const AirportCode_T & *iOffPoint*) [inline]

Set the off point.

Definition at line 148 of file LegDate.hpp.

References _offPoint.

32.96.4.19 void stdair::LegDate::setBoardingDate (const Date_T & *iBoardingDate*) [inline]

Set the boarding date.

Definition at line 153 of file LegDate.hpp.

References _boardingDate.

32.96.4.20 void stdair::LegDate::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 158 of file LegDate.hpp.

References _boardingTime.

32.96.4.21 void stdair::LegDate::setOffDate (const Date_T & *iOffDate*) [inline]

Set the off date.

Definition at line 163 of file LegDate.hpp.

References _offDate.

32.96.4.22 void stdair::LegDate::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 168 of file LegDate.hpp.

References _offTime.

32.96.4.23 void stdair::LegDate::setElapsedTime (const Duration_T & *iElapsedTime*)

Set the elapsed time.

Definition at line 103 of file LegDate.cpp.

References _elapsedTime.

32.96.4.24 void stdair::LegDate::setOperatingAirlineCode (const AirlineCode_T & *iAirlineCode*) [inline]

Set the operating airline code.

Definition at line 176 of file LegDate.hpp.

References _operatingAirlineCode.

32.96.4.25 void stdair::LegDate::setOperatingFlightNumber (const FlightNumber_T & *iFlightNumber*) [inline]

Set the operating flight number.

Definition at line 181 of file LegDate.hpp.

References _operatingFlightNumber.

32.96.4.26 void stdair::LegDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 194 of file LegDate.hpp.

References toString().

32.96.4.27 void stdair::LegDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 200 of file LegDate.hpp.

32.96.4.28 std::string stdair::LegDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 56 of file LegDate.cpp.

References describeKey().

Referenced by toStream().

32.96.4.29 const std::string stdair::LegDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 207 of file LegDate.hpp.

References _key, and stdair::LegDateKey::toString().

Referenced by describeRoutingKey(), stdair::LegCabin::getFullerKey(), and toString().

32.96.4.30 const std::string stdair::LegDate::describeRoutingKey () const

Get a string describing the routing key.

Definition at line 63 of file LegDate.cpp.

References _operatingAirlineCode, _operatingFlightNumber, stdair::DEFAULT_KEY_FLD_-DELIMITER, describeKey(), stdair::FlightDate::getDepartureDate(), and getParent().

32.96.5 Friends And Related Function Documentation

32.96.5.1 friend class FacBom [friend]

Definition at line 26 of file LegDate.hpp.

32.96.5.2 friend class FacCloneBom [friend]

Definition at line 27 of file LegDate.hpp.

32.96.5.3 friend class FacBomManager [friend]

Definition at line 28 of file LegDate.hpp.

32.96.6 Member Data Documentation

32.96.6.1 Key_T stdair::LegDate::_key [protected]

Primary key (origin airport).

Definition at line 231 of file LegDate.hpp.

Referenced by describeKey(), getBoardingPoint(), and getKey().

32.96.6.2 BomAbstract* stdair::LegDate::_parent [protected]

Pointer on the parent class (FlightDate).

Definition at line 234 of file LegDate.hpp.

Referenced by getParent().

32.96.6.3 HolderMap_T stdair::LegDate::_holderMap [protected]

Map holding the children (LegCabin objects).

Definition at line 237 of file LegDate.hpp.

Referenced by getHolderMap().

32.96.6.4 AirportCode_T stdair::LegDate::_offPoint [protected]

Landing airport.

Definition at line 240 of file LegDate.hpp.

Referenced by getOffPoint(), and setOffPoint().

32.96.6.5 Date_T stdair::LegDate::_boardingDate [protected]

Boarding date.

Definition at line 243 of file LegDate.hpp.

Referenced by getBoardingDate(), getDateOffset(), and setBoardingDate().

32.96.6.6 Duration_T stdair::LegDate::_boardingTime [protected]

Boarding time.

Definition at line 246 of file LegDate.hpp.

Referenced by getBoardingTime(), getTimeOffset(), and setBoardingTime().

32.96.6.7 Date_T stdair::LegDate::_offDate [protected]

Landing date.

Definition at line 249 of file LegDate.hpp.

Referenced by getDateOffset(), getOffDate(), and setOffDate().

32.96.6.8 Duration_T stdair::LegDate::_offTime [protected]

Landing time.

Definition at line 252 of file LegDate.hpp.

Referenced by getOffTime(), getTimeOffset(), and setOffTime().

32.96.6.9 Duration_T stdair::LegDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 255 of file LegDate.hpp.

Referenced by getElapsedTime(), getTimeOffset(), and setElapsedTime().

32.96.6.10 Distance_T stdair::LegDate::_distance [protected]

Trip distance.

Definition at line 258 of file LegDate.hpp.

Referenced by getDistance().

32.96.6.11 CabinCapacity_T stdair::LegDate::_capacity [protected]

Aggregated capacity for all the leg-cabins.

Definition at line 261 of file LegDate.hpp.

Referenced by getCapacity().

32.96.6.12 AirlineCode_T stdair::LegDate::_operatingAirlineCode [protected]

Operating airline code.

Definition at line 264 of file LegDate.hpp.

Referenced by describeRoutingKey(), and setOperatingAirlineCode().

32.96.6.13 FlightNumber_T stdair::LegDate::_operatingFlightNumber [protected]

Operating flight number.

Definition at line 267 of file LegDate.hpp.

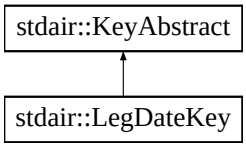
Referenced by describeRoutingKey(), and setOperatingFlightNumber().

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

- stdair/bom/LegDate.hpp
- stdair/bom/LegDate.cpp

32.97 stdair::LegDateKey Struct Reference

`#include <stdair/bom/LegDateKey.hpp>`Inheritance diagram for stdair::LegDateKey::



Public Member Functions

- LegDateKey (const AirportCode_T &iBoardingPoint)
- LegDateKey (const LegDateKey &)
- ~LegDateKey ()
- const AirportCode_T & getBoardingPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.97.1 Detailed Description

Key of a given leg-date, made of an origin airport.

Definition at line 16 of file LegDateKey.hpp.

32.97.2 Constructor & Destructor Documentation

32.97.2.1 stdair::LegDateKey::LegDateKey (const AirportCode_T & *iBoardingPoint*)

Constructor.

Definition at line 19 of file LegDateKey.cpp.

32.97.2.2 stdair::LegDateKey::LegDateKey (const LegDateKey & *iKey*)

Default copy constructor.

Definition at line 24 of file LegDateKey.cpp.

32.97.2.3 stdair::LegDateKey::~LegDateKey ()

Destructor.

Definition at line 29 of file LegDateKey.cpp.

32.97.3 Member Function Documentation

32.97.3.1 const AirportCode_T& stdair::LegDateKey::getBoardingPoint () const [inline]

Get the boarding point.

Definition at line 34 of file LegDateKey.hpp.

Referenced by stdair::LegDate::getBoardingPoint().

32.97.3.2 void stdair::LegDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 33 of file LegDateKey.cpp.

References toString().

32.97.3.3 void stdair::LegDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file LegDateKey.cpp.

32.97.3.4 const std::string stdair::LegDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same leg-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file LegDateKey.cpp.

Referenced by stdair::LegDate::describeKey(), stdair::FlightDate::getLegDate(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/LegDateKey.hpp
- stdair/bom/LegDateKey.cpp

32.98 stdair::Logger Class Reference

```
#include <stdair/service/Logger.hpp>
```

Public Member Functions

- template<typename T >
  void log (const LOG::EN_LogLevel iLevel, const int iLineNumber, const std::string &iFileName, const T &iToBeLogged)

Static Public Member Functions

- static Logger & instance ()

Friends

- class FacSupervisor

  Friend classes.

- class STDAIR_Service

32.98.1 Detailed Description

Class holding the stream for logs.

Note that the error logs are seen as standard output logs, but with a higher level of visibility.

Definition at line 48 of file Logger.hpp.

32.98.2 Member Function Documentation

32.98.2.1 template<typename T > void stdair::Logger::log (const LOG::EN_LogLevel *iLevel*, const int *iLineNumber*, const std::string & *iFileName*, const T & *iToBeLogged*) `[inline]`

Main log entry.

Definition at line 59 of file Logger.hpp.

References stdair::LOG::_logLevels.

32.98.2.2 Logger & stdair::Logger::instance () `[static]`

Return the static Logger instance.

Definition at line 48 of file Logger.cpp.

32.98.3 Friends And Related Function Documentation

32.98.3.1 friend class FacSupervisor `[friend]`

Friend classes.

Definition at line 50 of file Logger.hpp.

32.98.3.2 friend class STDAIR_Service [friend]

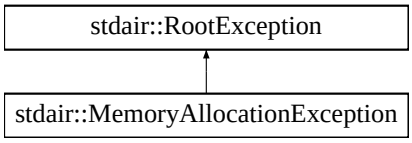
Definition at line 51 of file Logger.hpp.

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

- stdair/service/Logger.hpp
- stdair/service/Logger.cpp

32.99 stdair::MemoryAllocationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::MemoryAllocationException::



Public Member Functions

- MemoryAllocationException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.99.1 Detailed Description

Memory allocation.

Definition at line 89 of file stdair_exceptions.hpp.

32.99.2 Constructor & Destructor Documentation

32.99.2.1 stdair::MemoryAllocationException::MemoryAllocationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 92 of file stdair_exceptions.hpp.

32.99.3 Member Function Documentation

32.99.3.1 const char∗ stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.99.4 Member Data Documentation

32.99.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

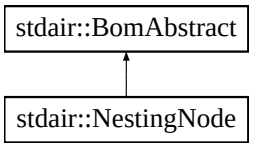
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.100 stdair::NestingNode Class Reference

`#include <stdair/bom/NestingNode.hpp>`Inheritance diagram for stdair::NestingNode::



Public Types

- typedef NestingNodeKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const Yield_T & getYield () const
- void setYield (const Yield_T &iYield)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- NestingNode (const Key_T &)
- virtual ~NestingNode ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

32.100.1 Detailed Description

Structure holding the elements of a nesting node. A nesting node is a set of booking classes.

Definition at line 29 of file NestingNode.hpp.

32.100.2 Member Typedef Documentation

32.100.2.1 typedef NestingNodeKey stdair::NestingNode::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 39 of file NestingNode.hpp.

32.100.3 Constructor & Destructor Documentation

32.100.3.1 stdair::NestingNode::NestingNode (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 31 of file NestingNode.cpp.

32.100.3.2 stdair::NestingNode::~NestingNode () `[protected, virtual]`

Destructor.

Definition at line 35 of file NestingNode.cpp.

32.100.4 Member Function Documentation

32.100.4.1 const Key_T& stdair::NestingNode::getKey () const `[inline]`

Get the policy key.

Definition at line 44 of file NestingNode.hpp.

32.100.4.2 BomAbstract* const stdair::NestingNode::getParent () const `[inline]`

Get the parent object.

Definition at line 49 of file NestingNode.hpp.

32.100.4.3 const HolderMap_T& stdair::NestingNode::getHolderMap () const `[inline]`

Get the map of children holders.

Definition at line 56 of file NestingNode.hpp.

Referenced by stdair::FacBomManager::resetYieldBasedNestingStructure().

32.100.4.4 const Yield_T& stdair::NestingNode::getYield () const [inline]

Getter for the yield.

Definition at line 61 of file NestingNode.hpp.

32.100.4.5 void stdair::NestingNode::setYield (const Yield_T & *iYield*) [inline]

Setter for the yield.

Definition at line 68 of file NestingNode.hpp.

Referenced by stdair::FacBomManager::resetYieldBasedNestingStructure().

32.100.4.6 void stdair::NestingNode::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 80 of file NestingNode.hpp.

References toString().

32.100.4.7 void stdair::NestingNode::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 89 of file NestingNode.hpp.

32.100.4.8 std::string stdair::NestingNode::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 39 of file NestingNode.cpp.

References describeKey().

Referenced by toStream().

32.100.4.9 const std::string stdair::NestingNode::describeKey () const [inline]

Get a string describing the key.

Definition at line 100 of file NestingNode.hpp.

References stdair::NestingNodeKey::toString().

Referenced by stdair::FacBomManager::resetYieldBasedNestingStructure(), and toString().

32.100.4.10 template<class Archive > void stdair::NestingNode::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

32.100.5 Friends And Related Function Documentation

32.100.5.1 friend class FacBom [friend]

Definition at line 30 of file NestingNode.hpp.

32.100.5.2 friend class FacBomManager [friend]

Definition at line 31 of file NestingNode.hpp.

32.100.5.3 friend class boost::serialization::access [friend]

Definition at line 32 of file NestingNode.hpp.

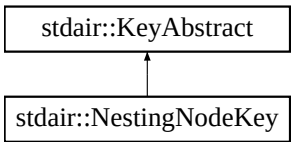
The documentation for this class was generated from the following files:

- stdair/bom/NestingNode.hpp
- stdair/bom/NestingNode.cpp

32.101 stdair::NestingNodeKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/NestingNodeKey.hpp>`Inheritance diagram for stdair::NestingNodeKey::



Public Member Functions

- NestingNodeKey (const NestingNodeCode_T &iNestingNodeCode)
- NestingNodeKey (const NestingNodeKey &)
- ~NestingNodeKey ()
- const NestingNodeCode_T & getNestingNodeCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.101.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file NestingNodeKey.hpp.

32.101.2 Constructor & Destructor Documentation

32.101.2.1 stdair::NestingNodeKey::NestingNodeKey (const NestingNodeCode_T & *iNestingNodeCode*)

Constructor.

Definition at line 28 of file NestingNodeKey.cpp.

32.101.2.2 stdair::NestingNodeKey::NestingNodeKey (const NestingNodeKey & *iNestingNodeKey*)

Copy constructor.

Definition at line 23 of file NestingNodeKey.cpp.

32.101.2.3 stdair::NestingNodeKey::~NestingNodeKey ()

Destructor.

Definition at line 33 of file NestingNodeKey.cpp.

32.101.3 Member Function Documentation

32.101.3.1 const NestingNodeCode_T& stdair::NestingNodeKey::getNestingNodeCode () const `[inline]`

Get the policy code.

Definition at line 56 of file NestingNodeKey.hpp.

32.101.3.2 void stdair::NestingNodeKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file NestingNodeKey.cpp.

References toString().

32.101.3.3 void stdair::NestingNodeKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file NestingNodeKey.cpp.

32.101.3.4 const std::string stdair::NestingNodeKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file NestingNodeKey.cpp.

Referenced by stdair::NestingNode::describeKey(), and toStream().

32.101.3.5 template<class Archive > void stdair::NestingNodeKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 68 of file NestingNodeKey.cpp.

32.101.4 Friends And Related Function Documentation

32.101.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file NestingNodeKey.hpp.

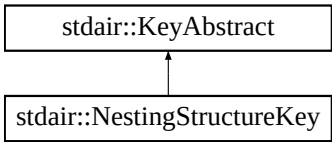
The documentation for this struct was generated from the following files:

- stdair/bom/NestingNodeKey.hpp
- stdair/bom/NestingNodeKey.cpp

32.102 stdair::NestingStructureKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/NestingStructureKey.hpp>`Inheritance diagram for stdair::NestingStructureKey::



Public Member Functions

- NestingStructureKey (const NestingStructureCode_T &iNestingStructureCode)
- NestingStructureKey (const NestingStructureKey &)
- ~NestingStructureKey ()
- const NestingStructureCode_T & getNestingStructureCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.102.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file NestingStructureKey.hpp.

32.102.2 Constructor & Destructor Documentation

32.102.2.1 stdair::NestingStructureKey::NestingStructureKey (const NestingStructureCode_T & *iNestingStructureCode*)

Constructor.

Definition at line 28 of file NestingStructureKey.cpp.

32.102.2.2 stdair::NestingStructureKey::NestingStructureKey (const NestingStructureKey & *iNestingStructureKey*)

Copy constructor.

Definition at line 23 of file NestingStructureKey.cpp.

32.102.2.3 stdair::NestingStructureKey::~NestingStructureKey ()

Destructor.

Definition at line 33 of file NestingStructureKey.cpp.

32.102.3 Member Function Documentation

32.102.3.1 const NestingStructureCode_T& stdair::NestingStructureKey::getNestingStructureCode () const [inline]

Get the nesting structure code.

Definition at line 56 of file NestingStructureKey.hpp.

32.102.3.2 void stdair::NestingStructureKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file NestingStructureKey.cpp.

References toString().

32.102.3.3 void stdair::NestingStructureKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file NestingStructureKey.cpp.

32.102.3.4 const std::string stdair::NestingStructureKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file NestingStructureKey.cpp.

Referenced by stdair::SimpleNestingStructure::describeKey(), and toStream().

32.102.3.5 template<class Archive > void stdair::NestingStructureKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 68 of file NestingStructureKey.cpp.

32.102.4 Friends And Related Function Documentation

32.102.4.1 friend class boost::serialization::access `[friend]`

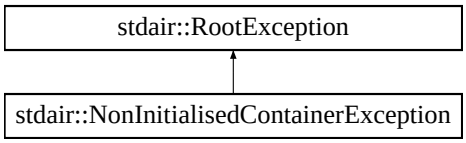
Definition at line 27 of file NestingStructureKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/NestingStructureKey.hpp
- stdair/bom/NestingStructureKey.cpp

32.103 stdair::NonInitialisedContainerException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::NonInitialisedContainerException::



Public Member Functions

- NonInitialisedContainerException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.103.1 Detailed Description

Non initialised container.

Definition at line 73 of file stdair_exceptions.hpp.

32.103.2 Constructor & Destructor Documentation

32.103.2.1 stdair::NonInitialisedContainerException::NonInitialisedContainerException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 76 of file stdair_exceptions.hpp.

32.103.3 Member Function Documentation

32.103.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.103.4 Member Data Documentation

32.103.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

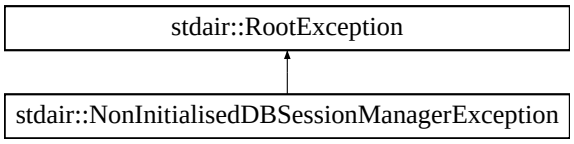
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.104 stdair::NonInitialisedDBSessionManagerException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::NonInitialisedDBSessionManagerException::



Public Member Functions

- NonInitialisedDBSessionManagerException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.104.1 Detailed Description

Non initialised database session.

Definition at line 188 of file stdair_exceptions.hpp.

32.104.2 Constructor & Destructor Documentation

32.104.2.1 stdair::NonInitialisedDBSessionManagerException::NonInitialisedDBSessionManagerException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 191 of file stdair_exceptions.hpp.

32.104.3 Member Function Documentation

32.104.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.104.4 Member Data Documentation

32.104.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

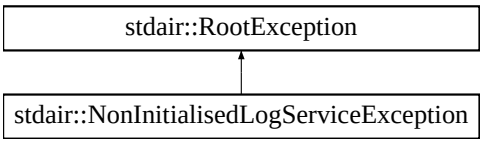
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.105 stdair::NonInitialisedLogServiceException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::NonInitialisedLogServiceException::



Public Member Functions

- NonInitialisedLogServiceException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.105.1 Detailed Description

Non initialised log service.

Definition at line 57 of file stdair_exceptions.hpp.

32.105.2 Constructor & Destructor Documentation

32.105.2.1 stdair::NonInitialisedLogServiceException::NonInitialisedLogServiceException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 60 of file stdair_exceptions.hpp.

32.105.3 Member Function Documentation

32.105.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.105.4 Member Data Documentation

32.105.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

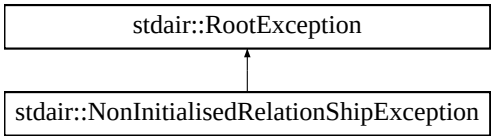
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.106 stdair::NonInitialisedRelationShipException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::NonInitialisedRelationShipException::



Public Member Functions

- NonInitialisedRelationShipException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.106.1 Detailed Description

Non initialised relationship.

Definition at line 81 of file stdair_exceptions.hpp.

32.106.2 Constructor & Destructor Documentation

32.106.2.1 stdair::NonInitialisedRelationShipException::NonInitialisedRelationShipException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 84 of file stdair_exceptions.hpp.

32.106.3 Member Function Documentation

32.106.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.106.4 Member Data Documentation

32.106.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

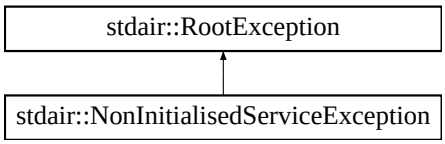
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.107 stdair::NonInitialisedServiceException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::NonInitialisedServiceException::



Public Member Functions

- NonInitialisedServiceException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.107.1 Detailed Description

Non initialised service.

Definition at line 65 of file stdair_exceptions.hpp.

32.107.2 Constructor & Destructor Documentation

32.107.2.1 stdair::NonInitialisedServiceException::NonInitialisedServiceException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 68 of file stdair_exceptions.hpp.

32.107.3 Member Function Documentation

32.107.3.1 const char∗ stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.107.4 Member Data Documentation

32.107.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

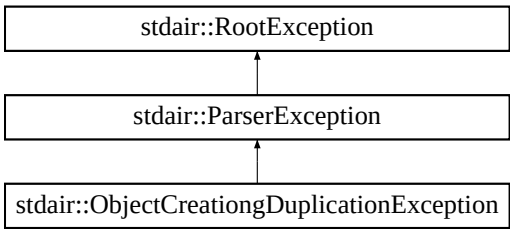
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.108 stdair::ObjectCreationgDuplicationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::ObjectCreationgDuplicationException::



Public Member Functions

- ObjectCreationgDuplicationException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.108.1 Detailed Description

Duplicated object.

Definition at line 157 of file stdair_exceptions.hpp.

32.108.2 Constructor & Destructor Documentation

32.108.2.1 stdair::ObjectCreationgDuplicationException::ObjectCreationgDuplicationException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 160 of file stdair_exceptions.hpp.

32.108.3 Member Function Documentation

32.108.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.108.4 Member Data Documentation

32.108.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

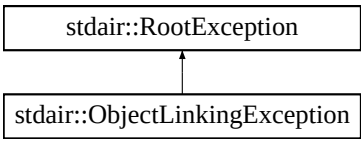
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.109 stdair::ObjectLinkingException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::ObjectLinkingException::



Public Member Functions

- ObjectLinkingException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.109.1 Detailed Description

Object link.

Definition at line 97 of file stdair_exceptions.hpp.

32.109.2 Constructor & Destructor Documentation

32.109.2.1 stdair::ObjectLinkingException::ObjectLinkingException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 100 of file stdair_exceptions.hpp.

32.109.3 Member Function Documentation

32.109.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.109.4 Member Data Documentation

32.109.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

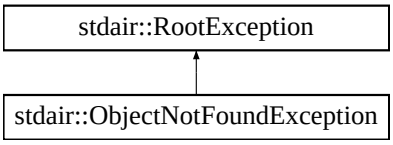
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.110 stdair::ObjectNotFoundException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::ObjectNotFoundException::



Public Member Functions

- ObjectNotFoundException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.110.1 Detailed Description

Not found object.

Definition at line 165 of file stdair_exceptions.hpp.

32.110.2 Constructor & Destructor Documentation

32.110.2.1 stdair::ObjectNotFoundException::ObjectNotFoundException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 168 of file stdair_exceptions.hpp.

32.110.3 Member Function Documentation

32.110.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.110.4 Member Data Documentation

32.110.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

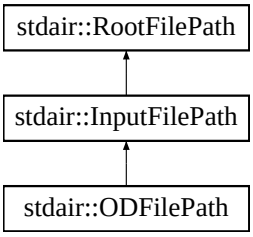
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.111 stdair::ODFilePath Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::ODFilePath::



Public Member Functions

- ODFilePath (const Filename_T &iFilename)
- const char ∗ name () const

Protected Attributes

- const Filename_T _filename

32.111.1 Detailed Description

OD input file.

Definition at line 76 of file stdair_file.hpp.

32.111.2 Constructor & Destructor Documentation

32.111.2.1 stdair::ODFilePath::ODFilePath (const Filename_T & *iFilename*) `[inline, explicit]`

Constructor.

Definition at line 81 of file stdair_file.hpp.

32.111.3 Member Function Documentation

32.111.3.1 const char∗ stdair::RootFilePath::name () const `[inline, inherited]`

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.111.4 Member Data Documentation

32.111.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

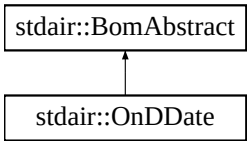
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

32.112 stdair::OnDDate Class Reference

Class representing the actual attributes for an airline flight-date.

`#include <stdair/bom/OnDDate.hpp>`Inheritance diagram for stdair::OnDDate::



Public Types

- typedef OnDDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirlineCode_T & getAirlineCode () const
- const stdair::Date_T getDate () const
- const stdair::AirportCode_T getOrigin () const
- const stdair::AirportCode_T getDestination () const
- const HolderMap_T & getHolderMap () const
- const StringDemandStructMap_T & getDemandInfoMap () const
- const CabinForecastMap_T & getTotalForecastMap () const
- const WTPDemandPair_T & getTotalForecast (const CabinCode_T &iCC) const
- const CabinClassPairList_T & getCabinClassPairList (const std::string &iStr) const
- const short getNbOfSegments () const
- void setDemandInformation (const CabinClassPairList_T &, const YieldDemandPair_T &)
- void setTotalForecast (const CabinCode_T &, const WTPDemandPair_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- OnDDate (const Key_T &)
- virtual ~OnDDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- StringDemandStructMap_T _classPathDemandMap
- StringCabinClassPairListMap_T _stringCabinClassPairListMap
- CabinForecastMap_T _cabinForecastMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.112.1 Detailed Description

Class representing the actual attributes for an airline flight-date.

Definition at line 33 of file OnDDate.hpp.

32.112.2 Member Typedef Documentation

32.112.2.1 typedef OnDDateKey stdair::OnDDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file OnDDate.hpp.

32.112.3 Constructor & Destructor Documentation

32.112.3.1 stdair::OnDDate::OnDDate (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 28 of file OnDDate.cpp.

32.112.3.2 stdair::OnDDate::~OnDDate () `[protected, virtual]`

Destructor.

Definition at line 33 of file OnDDate.cpp.

32.112.4 Member Function Documentation

32.112.4.1 const Key_T& stdair::OnDDate::getKey () const [inline]

Get the O&D date key.

Definition at line 50 of file OnDDate.hpp.

References _key.

32.112.4.2 BomAbstract* const stdair::OnDDate::getParent () const [inline]

Get the parent object.

Definition at line 55 of file OnDDate.hpp.

References _parent.

Referenced by getAirlineCode().

32.112.4.3 const AirlineCode_T & stdair::OnDDate::getAirlineCode () const

Get the airline code (key of the parent object).

Note:

That method assumes that the parent object derives from the Inventory class, as it needs to have access to the getAirlineCode() method.

Definition at line 44 of file OnDDate.cpp.

References stdair::Inventory::getAirlineCode(), and getParent().

32.112.4.4 const stdair::Date_T stdair::OnDDate::getDate () const [inline]

Get the boarding date.

Definition at line 70 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getDate().

32.112.4.5 const stdair::AirportCode_T stdair::OnDDate::getOrigin () const [inline]

Get the origin.

Definition at line 75 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getOrigin().

32.112.4.6 const stdair::AirportCode_T stdair::OnDDate::getDestination () const [inline]

Get the destination.

Definition at line 80 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getDestination().

32.112.4.7 const HolderMap_T& stdair::OnDDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 87 of file OnDDate.hpp.

References _holderMap.

32.112.4.8 const StringDemandStructMap_T& stdair::OnDDate::getDemandInfoMap () const [inline]

Get the map of demand information.

Definition at line 94 of file OnDDate.hpp.

References _classPathDemandMap.

32.112.4.9 const CabinForecastMap_T& stdair::OnDDate::getTotalForecastMap () const [inline]

Get the map of total forecast.

Definition at line 101 of file OnDDate.hpp.

References _cabinForecastMap.

32.112.4.10 const WTPDemandPair_T& stdair::OnDDate::getTotalForecast (const CabinCode_T & *iCC*) const [inline]

Get the total forecast for a given cabin.

Definition at line 108 of file OnDDate.hpp.

References _cabinForecastMap.

32.112.4.11 const CabinClassPairList_T& stdair::OnDDate::getCabinClassPairList (const std::string & *iStr*) const [inline]

Get the cabin-class pair out of a string.

Definition at line 116 of file OnDDate.hpp.

References _stringCabinClassPairListMap.

32.112.4.12 const short stdair::OnDDate::getNbOfSegments () const [inline]

Get the number of segments of the O&D.

Definition at line 124 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::getNbOfSegments().

32.112.4.13 void stdair::OnDDate::setDemandInformation (const CabinClassPairList_T & *iCabinClassPairList*, const YieldDemandPair_T & *iYieldDemandPair*)

Set demand information.

Definition at line 53 of file OnDDate.cpp.

References _classPathDemandMap, and _stringCabinClassPairListMap.

32.112.4.14 void stdair::OnDDate::setTotalForecast (const CabinCode_T & *iCabinCode*, const WTPDemandPair_T & *iWTPDemandPair*)

Set forecast information per cabin.

Definition at line 76 of file OnDDate.cpp.

References _cabinForecastMap.

32.112.4.15 void stdair::OnDDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 147 of file OnDDate.hpp.

References toString().

32.112.4.16 void stdair::OnDDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 156 of file OnDDate.hpp.

32.112.4.17 std::string stdair::OnDDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file OnDDate.cpp.

References describeKey().

Referenced by toStream().

32.112.4.18 const std::string stdair::OnDDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 167 of file OnDDate.hpp.

References _key, and stdair::OnDDateKey::toString().

Referenced by toString().

32.112.4.19 template<class Archive > void stdair::OnDDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

32.112.5 Friends And Related Function Documentation

32.112.5.1 friend class FacBom [friend]

Definition at line 34 of file OnDDate.hpp.

32.112.5.2 friend class FacCloneBom [friend]

Definition at line 35 of file OnDDate.hpp.

32.112.5.3 friend class FacBomManager [friend]

Definition at line 36 of file OnDDate.hpp.

32.112.5.4 friend class boost::serialization::access [friend]

Definition at line 37 of file OnDDate.hpp.

32.112.6 Member Data Documentation

32.112.6.1 Key_T stdair::OnDDate::_key [protected]

Primary key (list of OnD string keys).

Definition at line 217 of file OnDDate.hpp.

Referenced by describeKey(), getDate(), getDestination(), getKey(), getNbOfSegments(), and getOrigin().

32.112.6.2 BomAbstract* stdair::OnDDate::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 222 of file OnDDate.hpp.

Referenced by getParent().

32.112.6.3 HolderMap_T stdair::OnDDate::_holderMap [protected]

Map holding the children (SegmentDate and LegDate objects).

Definition at line 227 of file OnDDate.hpp.

Referenced by getHolderMap().

32.112.6.4 StringDemandStructMap_T stdair::OnDDate::_classPathDemandMap [protected]

O&D demand information.

Definition at line 232 of file OnDDate.hpp.

Referenced by getDemandInfoMap(), and setDemandInformation().

32.112.6.5 StringCabinClassPairListMap_T stdair::OnDDate::_stringCabinClassPairListMap [protected]

O&D cabin and associated class map.

Definition at line 237 of file OnDDate.hpp.

Referenced by getCabinClassPairList(), and setDemandInformation().

32.112.6.6 CabinForecastMap_T stdair::OnDDate::_cabinForecastMap [protected]

O&D demand total forecast.

Definition at line 242 of file OnDDate.hpp.

Referenced by getTotalForecast(), getTotalForecastMap(), and setTotalForecast().

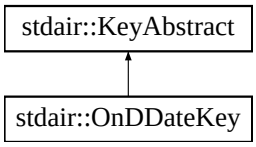
The documentation for this class was generated from the following files:

- stdair/bom/OnDDate.hpp
- stdair/bom/OnDDate.cpp

32.113 stdair::OnDDateKey Struct Reference

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

`#include <stdair/bom/OnDDateKey.hpp>`Inheritance diagram for stdair::OnDDateKey::



Public Member Functions

- OnDDateKey (const OnDStringList_T &)

- OnDDateKey (const OnDDateKey &)
- ~OnDDateKey ()
- const Date_T getDate () const
- const AirportCode_T getOrigin () const
- const AirportCode_T getDestination () const
- const short getNbOfSegments () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.113.1 Detailed Description

Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

Definition at line 23 of file OnDDateKey.hpp.

32.113.2 Constructor & Destructor Documentation

32.113.2.1 stdair::OnDDateKey::OnDDateKey (const OnDStringList_T & *iOnDStringList*)

Constructor.

Definition at line 33 of file OnDDateKey.cpp.

32.113.2.2 stdair::OnDDateKey::OnDDateKey (const OnDDateKey & *iKey*)

Copy constructor.

Definition at line 38 of file OnDDateKey.cpp.

32.113.2.3 stdair::OnDDateKey::~OnDDateKey ()

Destructor.

Definition at line 43 of file OnDDateKey.cpp.

32.113.3 Member Function Documentation

32.113.3.1 const Date_T stdair::OnDDateKey::getDate () const

Get the boarding date.

Definition at line 47 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractFlightDateKey(), and stdair::FlightDateKey::getDepartureDate().

Referenced by stdair::OnDDate::getDate().

32.113.3.2 const AirportCode_T stdair::OnDDateKey::getOrigin () const

Get the origin.

Definition at line 54 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::SegmentDateKey::getBoardingPoint().

Referenced by stdair::OnDDate::getOrigin().

32.113.3.3 const AirportCode_T stdair::OnDDateKey::getDestination () const

Get the destination.

Definition at line 61 of file OnDDateKey.cpp.

References stdair::BomKeyManager::extractSegmentDateKey(), and stdair::SegmentDateKey::getOffPoint().

Referenced by stdair::OnDDate::getDestination().

32.113.3.4 const short stdair::OnDDateKey::getNbOfSegments () const `[inline]`

Get the number of segments.

Definition at line 70 of file OnDDateKey.hpp.

Referenced by stdair::OnDDate::getNbOfSegments().

32.113.3.5 void stdair::OnDDateKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 68 of file OnDDateKey.cpp.

References toString().

32.113.3.6 void stdair::OnDDateKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 73 of file OnDDateKey.cpp.

32.113.3.7 const std::string stdair::OnDDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 77 of file OnDDateKey.cpp.

Referenced by stdair::OnDDate::describeKey(), and toStream().

32.113.3.8 template<class Archive > void stdair::OnDDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 102 of file OnDDateKey.cpp.

32.113.4 Friends And Related Function Documentation

32.113.4.1 friend class boost::serialization::access [friend]

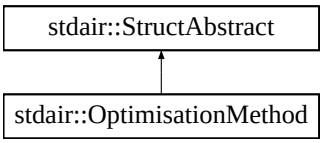
Definition at line 24 of file OnDDateKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/OnDDateKey.hpp
- stdair/bom/OnDDateKey.cpp

32.114 stdair::OptimisationMethod Struct Reference

`#include <stdair/basic/OptimisationMethod.hpp>`Inheritance diagram for stdair::OptimisationMethod::



Public Types

- enum EN_OptimisationMethod { LEG_BASED_MC = 0, LEG_BASED_EMSR_B, LAST_VALUE }

Public Member Functions

- EN_OptimisationMethod getMethod () const

- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_OptimisationMethod &) const
- OptimisationMethod (const EN_OptimisationMethod &)
- OptimisationMethod (const char iMethod)
- OptimisationMethod (const OptimisationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_OptimisationMethod &)
- static char getMethodLabel (const EN_OptimisationMethod &)
- static std::string getMethodLabelAsString (const EN_OptimisationMethod &)
- static std::string describeLabels ()

32.114.1 Detailed Description

Enumeration of Optimisation methods.

Definition at line 15 of file OptimisationMethod.hpp.

32.114.2 Member Enumeration Documentation

32.114.2.1 enum stdair::OptimisationMethod::EN_OptimisationMethod

Enumerator:

LEG_BASED_MC

LEG_BASED_EMSR_B

LAST_VALUE

Definition at line 17 of file OptimisationMethod.hpp.

32.114.3 Constructor & Destructor Documentation

32.114.3.1 stdair::OptimisationMethod::OptimisationMethod (const EN_OptimisationMethod & *iOptimisationMethod*)

Constructor.

Definition at line 36 of file OptimisationMethod.cpp.

32.114.3.2 stdair::OptimisationMethod::OptimisationMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file OptimisationMethod.cpp.

References describeLabels(), LAST_VALUE, LEG_BASED_EMSR_B, and LEG_BASED_MC.

32.114.3.3 stdair::OptimisationMethod::OptimisationMethod (const OptimisationMethod & *iOptimisationMethod*)

Default copy constructor.

Definition at line 30 of file OptimisationMethod.cpp.

32.114.4 Member Function Documentation

32.114.4.1 const std::string & stdair::OptimisationMethod::getLabel (const EN_OptimisationMethod & *iMethod*) `[static]`

Get the label as a string (e.g., "Leg based Monte Carlo" or "Leg based EMSRb").

Definition at line 59 of file OptimisationMethod.cpp.

32.114.4.2 char stdair::OptimisationMethod::getMethodLabel (const EN_OptimisationMethod & *iMethod*) `[static]`

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 64 of file OptimisationMethod.cpp.

32.114.4.3 std::string stdair::OptimisationMethod::getMethodLabelAsString (const EN_OptimisationMethod & *iMethod*) `[static]`

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 70 of file OptimisationMethod.cpp.

32.114.4.4 std::string stdair::OptimisationMethod::describeLabels () `[static]`

List the labels.

Definition at line 77 of file OptimisationMethod.cpp.

References LAST_VALUE.

Referenced by OptimisationMethod().

32.114.4.5 OptimisationMethod::EN_OptimisationMethod stdair::OptimisationMethod::getMethod () const

Get the enumerated value.

Definition at line 89 of file OptimisationMethod.cpp.

Referenced by stdair::AirlineFeature::getOptimisationMethod().

32.114.4.6 std::string stdair::OptimisationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "M" or "E").

Definition at line 94 of file OptimisationMethod.cpp.

32.114.4.7 const std::string stdair::OptimisationMethod::describe () const [virtual]

Give a description of the structure (e.g., "Leg based Monte Carlo" or "Leg based EMSRb").

Implements stdair::StructAbstract.

Definition at line 101 of file OptimisationMethod.cpp.

32.114.4.8 bool stdair::OptimisationMethod::operator== (const EN_OptimisationMethod & *iMethod*) const

Comparaison operator.

Definition at line 109 of file OptimisationMethod.cpp.

32.114.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.114.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

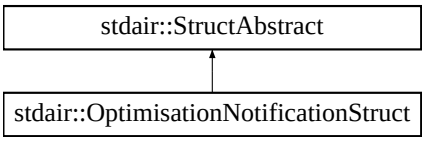
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/OptimisationMethod.hpp
- stdair/basic/OptimisationMethod.cpp

32.115 stdair::OptimisationNotificationStruct Struct Reference

`#include <stdair/bom/OptimisationNotificationStruct.hpp>`Inheritance diagram for stdair::OptimisationNotificationStruct::



Public Member Functions

- const AirportCode_T & getOrigin () const
- const AirportCode_T & getDestination () const
- const CityCode_T & getPOS () const
- const Date_T & getPreferedDepartureDate () const
- const DateTime_T & getNotificationDateTime () const
- const CabinCode_T & getPreferredCabin () const
- const NbOfSeats_T & getPartySize () const
- const ChannelLabel_T & getOptimisationChannel () const
- const TripType_T & getTripType () const
- const DayDuration_T & getStayDuration () const
- const FrequentFlyer_T & getFrequentFlyerType () const
- const Duration_T & getPreferredDepartureTime () const
- const WTP_T & getWTP () const
- const PriceValue_T & getValueOfTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- OptimisationNotificationStruct (const AirportCode_T &iOrigin, const AirportCode_T &iDestination, const CityCode_T &iPOS, const Date_T &iDepartureDate, const DateTime_T &iNotificationDateTime, const CabinCode_T &iPreferredCabin, const NbOfSeats_T &iPartySize, const ChannelLabel_T &iChannel, const TripType_T &iTripType, const DayDuration_T &iStayDuration, const FrequentFlyer_T &iFrequentFlyerType, const Duration_T &iPreferredDepartureTime, const WTP_T &iWTP, const PriceValue_T &iValueOfTime)
- OptimisationNotificationStruct (const OptimisationNotificationStruct &)
- ~OptimisationNotificationStruct ()

32.115.1 Detailed Description

Structure holding the elements of a optimisation notification.

Definition at line 19 of file OptimisationNotificationStruct.hpp.

32.115.2 Constructor & Destructor Documentation

32.115.2.1 stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const CityCode_T & *iPOS*, const Date_T & *iDepartureDate*, const DateTime_T & *iNotificationDateTime*, const CabinCode_T & *iPreferredCabin*, const NbOfSeats_T & *iPartySize*, const ChannelLabel_T & *iChannel*, const TripType_T & *iTripType*, const DayDuration_T & *iStayDuration*, const FrequentFlyer_T & *iFrequentFlyerType*, const Duration_T & *iPreferredDepartureTime*, const WTP_T & *iWTP*, const PriceValue_T & *iValueOfTime*)

Constructor.

Definition at line 39 of file OptimisationNotificationStruct.cpp.

32.115.2.2 stdair::OptimisationNotificationStruct::OptimisationNotificationStruct (const OptimisationNotificationStruct & *iOptimisationNotification*)

Copy constructor.

Definition at line 20 of file OptimisationNotificationStruct.cpp.

32.115.2.3 stdair::OptimisationNotificationStruct::~OptimisationNotificationStruct ()

Destructor.

Definition at line 64 of file OptimisationNotificationStruct.cpp.

32.115.3 Member Function Documentation

32.115.3.1 const AirportCode_T& stdair::OptimisationNotificationStruct::getOrigin () const `[inline]`

Get the notificationed origin.

Definition at line 23 of file OptimisationNotificationStruct.hpp.

32.115.3.2 const AirportCode_T& stdair::OptimisationNotificationStruct::getDestination () const `[inline]`

Get the notificationed destination.

Definition at line 28 of file OptimisationNotificationStruct.hpp.

32.115.3.3 const CityCode_T& stdair::OptimisationNotificationStruct::getPOS () const `[inline]`

Get the point-of-sale.

Definition at line 33 of file OptimisationNotificationStruct.hpp.

32.115.3.4 const Date_T& stdair::OptimisationNotificationStruct::getPreferedDepartureDate () const [inline]

Get the notificationed departure date.

Definition at line 38 of file OptimisationNotificationStruct.hpp.

32.115.3.5 const DateTime_T& stdair::OptimisationNotificationStruct::getNotificationDateTime () const [inline]

Get the notification datetime.

Definition at line 43 of file OptimisationNotificationStruct.hpp.

32.115.3.6 const CabinCode_T& stdair::OptimisationNotificationStruct::getPreferredCabin () const [inline]

Get the preferred cabin.

Definition at line 48 of file OptimisationNotificationStruct.hpp.

32.115.3.7 const NbOfSeats_T& stdair::OptimisationNotificationStruct::getPartySize () const [inline]

Get the party size.

Definition at line 53 of file OptimisationNotificationStruct.hpp.

32.115.3.8 const ChannelLabel_T& stdair::OptimisationNotificationStruct::getOptimisationChannel () const [inline]

Get the reservation channel.

Definition at line 58 of file OptimisationNotificationStruct.hpp.

32.115.3.9 const TripType_T& stdair::OptimisationNotificationStruct::getTripType () const [inline]

Get the trip type.

Definition at line 63 of file OptimisationNotificationStruct.hpp.

32.115.3.10 const DayDuration_T& stdair::OptimisationNotificationStruct::getStayDuration () const [inline]

Get the duration of stay.

Definition at line 68 of file OptimisationNotificationStruct.hpp.

32.115.3.11 const FrequentFlyer_T& stdair::OptimisationNotificationStruct::getFrequentFlyerType () const [inline]

Get the frequent flyer type.

Definition at line 73 of file OptimisationNotificationStruct.hpp.

32.115.3.12 const Duration_T& stdair::OptimisationNotificationStruct::getPreferredDepartureTime () const [inline]

Get the preferred departure time.

Definition at line 78 of file OptimisationNotificationStruct.hpp.

32.115.3.13 const WTP_T& stdair::OptimisationNotificationStruct::getWTP () const [inline]

Get the willingness-to-pay.

Definition at line 83 of file OptimisationNotificationStruct.hpp.

32.115.3.14 const PriceValue_T& stdair::OptimisationNotificationStruct::getValueOfTime () const [inline]

Get the value of time.

Definition at line 88 of file OptimisationNotificationStruct.hpp.

32.115.3.15 void stdair::OptimisationNotificationStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 68 of file OptimisationNotificationStruct.cpp.

References describe().

32.115.3.16 void stdair::OptimisationNotificationStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 73 of file OptimisationNotificationStruct.cpp.

32.115.3.17 const std::string stdair::OptimisationNotificationStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 77 of file OptimisationNotificationStruct.cpp.

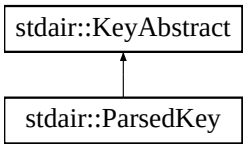
Referenced by toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/OptimisationNotificationStruct.hpp
- stdair/bom/OptimisationNotificationStruct.cpp

32.116 stdair::ParsedKey Struct Reference

`#include <stdair/bom/ParsedKey.hpp>`Inheritance diagram for stdair::ParsedKey::



Public Member Functions

- InventoryKey getInventoryKey () const
- FlightDateKey getFlightDateKey () const
- SegmentDateKey getSegmentKey () const
- LegDateKey getLegKey () const
- const Duration_T getBoardingTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- ParsedKey ()
- ~ParsedKey ()

Public Attributes

- std::string _fullKey
- std::string _airlineCode
- std::string _flightNumber
- std::string _departureDate
- std::string _boardingPoint
- std::string _offPoint
- std::string _boardingTime

32.116.1 Detailed Description

Structure which holds the results/keys after the parsing.

Definition at line 22 of file ParsedKey.hpp.

32.116.2 Constructor & Destructor Documentation

32.116.2.1 stdair::ParsedKey::ParsedKey ()

Definition at line 41 of file ParsedKey.cpp.

32.116.2.2 stdair::ParsedKey::~ParsedKey ()

Definition at line 47 of file ParsedKey.cpp.

32.116.3 Member Function Documentation

32.116.3.1 InventoryKey stdair::ParsedKey::getInventoryKey () const

Inventory key.

Definition at line 51 of file ParsedKey.cpp.

References _airlineCode, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractInventoryKey().

32.116.3.2 FlightDateKey stdair::ParsedKey::getFlightDateKey () const

Flight-date key.

Definition at line 62 of file ParsedKey.cpp.

References _departureDate, _flightNumber, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, stdair::TokeniserDashSeparator(), and toString().

Referenced by stdair::BomKeyManager::extractFlightDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.116.3.3 SegmentDateKey stdair::ParsedKey::getSegmentKey () const

Segment-date key.

Definition at line 98 of file ParsedKey.cpp.

References _boardingPoint, _fullKey, _offPoint, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractSegmentDateKey(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

32.116.3.4 LegDateKey stdair::ParsedKey::getLegKey () const

Leg-date key.

Definition at line 84 of file ParsedKey.cpp.

References _boardingPoint, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, and toString().

Referenced by stdair::BomKeyManager::extractLegDateKey().

32.116.3.5 const Duration_T stdair::ParsedKey::getBoardingTime () const

Boarding time.

Definition at line 112 of file ParsedKey.cpp.

References _boardingTime, _fullKey, STDAIR_LOG_DEBUG, STDAIR_LOG_ERROR, stdair::TokeniserTimeSeparator(), and toString().

32.116.3.6 void stdair::ParsedKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 130 of file ParsedKey.cpp.

References toString().

32.116.3.7 void stdair::ParsedKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 135 of file ParsedKey.cpp.

32.116.3.8 const std::string stdair::ParsedKey::toString () const `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 139 of file ParsedKey.cpp.

References _airlineCode, _boardingPoint, _boardingTime, _departureDate, _flightNumber, _offPoint, stdair::DEFAULT_KEY_FLD_DELIMITER, and stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::TravelSolutionStruct::describe(), stdair::TravelSolutionStruct::describeSegmentPath(), stdair::TravelSolutionStruct::display(), getBoardingTime(), getFlightDateKey(), getInventoryKey(), getLegKey(), getSegmentKey(), and toStream().

32.116.4 Member Data Documentation

32.116.4.1 std::string stdair::ParsedKey::_fullKey

Definition at line 76 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getBoardingTime(), getFlightDateKey(), getInventoryKey(), getLegKey(), and getSegmentKey().

32.116.4.2 std::string stdair::ParsedKey::_airlineCode

Definition at line 77 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getInventoryKey(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toString().

32.116.4.3 std::string stdair::ParsedKey::_flightNumber

Definition at line 78 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getFlightDateKey(), and toString().

32.116.4.4 std::string stdair::ParsedKey::_departureDate

Definition at line 79 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getFlightDateKey(), and toString().

32.116.4.5 std::string stdair::ParsedKey::_boardingPoint

Definition at line 80 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getLegKey(), getSegmentKey(), and toString().

32.116.4.6 std::string stdair::ParsedKey::_offPoint

Definition at line 81 of file ParsedKey.hpp.

Referenced by stdair::BomKeyManager::extractKeys(), getSegmentKey(), and toString().

32.116.4.7 std::string stdair::ParsedKey::_boardingTime

Definition at line 82 of file ParsedKey.hpp.

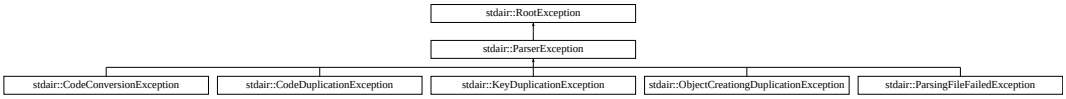
Referenced by stdair::BomKeyManager::extractKeys(), getBoardingTime(), and toString().

The documentation for this struct was generated from the following files:

- stdair/bom/ParsedKey.hpp
- stdair/bom/ParsedKey.cpp

32.117 stdair::ParserException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::ParserException::



Public Member Functions

- ParserException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.117.1 Detailed Description

Parser.

Definition at line 112 of file stdair_exceptions.hpp.

32.117.2 Constructor & Destructor Documentation

32.117.2.1 stdair::ParserException::ParserException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 115 of file stdair_exceptions.hpp.

32.117.3 Member Function Documentation

32.117.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.117.4 Member Data Documentation

32.117.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

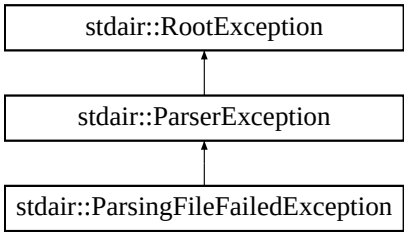
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.118 stdair::ParsingFileFailedException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::ParsingFileFailedException::



Public Member Functions

- ParsingFileFailedException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.118.1 Detailed Description

Input file parsing failure.

Definition at line 173 of file stdair_exceptions.hpp.

32.118.2 Constructor & Destructor Documentation

32.118.2.1 stdair::ParsingFileFailedException::ParsingFileFailedException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 176 of file stdair_exceptions.hpp.

32.118.3 Member Function Documentation

32.118.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.118.4 Member Data Documentation

32.118.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

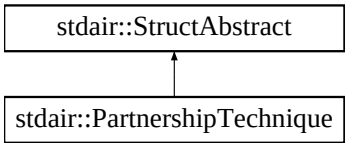
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

32.119 stdair::PartnershipTechnique Struct Reference

Enumeration of partnership techniques.

`#include <stdair/basic/PartnershipTechnique.hpp>`Inheritance diagram for stdair::PartnershipTechnique::



Public Types

- enum EN_PartnershipTechnique {
  NONE = 0, RAE_DA, RAE_YP, IBP_DA,
  IBP_YP, IBP_YP_U, RMC, A_RMC,
  LAST_VALUE }

Public Member Functions

- EN_PartnershipTechnique getTechnique () const
- char getTechniqueAsChar () const
- std::string getTechniqueAsString () const
- const std::string describe () const
- bool operator== (const EN_PartnershipTechnique &) const
- PartnershipTechnique (const EN_PartnershipTechnique &)
- PartnershipTechnique (const char iTechnique)
- PartnershipTechnique (const std::string &iTechnique)
- PartnershipTechnique (const PartnershipTechnique &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PartnershipTechnique &)
- static EN_PartnershipTechnique getTechnique (const char)
- static char getTechniqueLabel (const EN_PartnershipTechnique &)
- static std::string getTechniqueLabelAsString (const EN_PartnershipTechnique &)
- static std::string describeLabels ()

32.119.1 Detailed Description

Enumeration of partnership techniques.

Definition at line 17 of file PartnershipTechnique.hpp.

32.119.2 Member Enumeration Documentation

32.119.2.1 enum stdair::PartnershipTechnique::EN_PartnershipTechnique

Enumerator:

NONE

RAE_DA

RAE_YP

IBP_DA

IBP_YP

IBP_YP_U

RMC

A_RMC

LAST_VALUE

Definition at line 19 of file PartnershipTechnique.hpp.

32.119.3 Constructor & Destructor Documentation

32.119.3.1 stdair::PartnershipTechnique::PartnershipTechnique (const EN_PartnershipTechnique & *iPartnershipTechnique*)

Main constructor.

Definition at line 48 of file PartnershipTechnique.cpp.

32.119.3.2 stdair::PartnershipTechnique::PartnershipTechnique (const char *iTechnique*)

Alternative constructor.

Definition at line 82 of file PartnershipTechnique.cpp.

32.119.3.3 stdair::PartnershipTechnique::PartnershipTechnique (const std::string & *iTechnique*)

Alternative constructor.

Definition at line 88 of file PartnershipTechnique.cpp.

References getTechnique().

32.119.3.4 stdair::PartnershipTechnique::PartnershipTechnique (const PartnershipTechnique & *iPartnershipTechnique*)

Default copy constructor.

Definition at line 42 of file PartnershipTechnique.cpp.

32.119.4 Member Function Documentation

32.119.4.1 const std::string & stdair::PartnershipTechnique::getLabel (const EN_PartnershipTechnique & *iTechnique*) `[static]`

Get the label as a string (e.g., "RevenueManagementCooperation").

Definition at line 98 of file PartnershipTechnique.cpp.

32.119.4.2 PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique (const char *iTechniqueChar*) `[static]`

Get the technique value from parsing a single char (e.g., 'r' or 'C').

Definition at line 54 of file PartnershipTechnique.cpp.

References A_RMC, describeLabels(), IBP_DA, IBP_YP, IBP_YP_U, LAST_VALUE, NONE, RAE_DA, RAE_YP, and RMC.

Referenced by stdair::AirlineFeature::getPartnershipTechnique().

32.119.4.3 char stdair::PartnershipTechnique::getTechniqueLabel (const EN_PartnershipTechnique & *iTechnique*) [static]

Get the label as a single char (e.g., 'r' or 'C').

Definition at line 104 of file PartnershipTechnique.cpp.

32.119.4.4 std::string stdair::PartnershipTechnique::getTechniqueLabelAsString (const EN_PartnershipTechnique & *iTechnique*) [static]

Get the label as a string of a single char (e.g., "r" or "C").

Definition at line 110 of file PartnershipTechnique.cpp.

32.119.4.5 std::string stdair::PartnershipTechnique::describeLabels () [static]

List the labels.

Definition at line 117 of file PartnershipTechnique.cpp.

References LAST_VALUE.

Referenced by getTechnique().

32.119.4.6 PartnershipTechnique::EN_PartnershipTechnique stdair::PartnershipTechnique::getTechnique () const

Get the enumerated value.

Definition at line 130 of file PartnershipTechnique.cpp.

Referenced by PartnershipTechnique().

32.119.4.7 char stdair::PartnershipTechnique::getTechniqueAsChar () const

Get the enumerated value as a char (e.g., 'r' or 'C').

Definition at line 135 of file PartnershipTechnique.cpp.

32.119.4.8 std::string stdair::PartnershipTechnique::getTechniqueAsString () const

Get the enumerated value as a short string (e.g., "r" or "C").

Definition at line 141 of file PartnershipTechnique.cpp.

32.119.4.9 const std::string stdair::PartnershipTechnique::describe () const [virtual]

Give a description of the structure (e.g., "RevenueManagementCooperation" or "InterlineBidPriceYieldProration").

Implements stdair::StructAbstract.

Definition at line 148 of file PartnershipTechnique.cpp.

32.119.4.10 bool stdair::PartnershipTechnique::operator== (const EN_PartnershipTechnique & *iTechnique*) const

Comparison operator.

Definition at line 156 of file PartnershipTechnique.cpp.

32.119.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.119.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

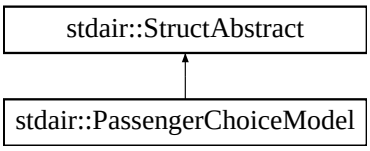
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/PartnershipTechnique.hpp
- stdair/basic/PartnershipTechnique.cpp

32.120 stdair::PassengerChoiceModel Struct Reference

`#include <stdair/basic/PassengerChoiceModel.hpp>`Inheritance diagram for stdair::PassengerChoiceModel::



Public Types

- enum EN_PassengerChoiceModel { HARD_RESTRICTION = 0, PRICE_ORIENTED, HYBRID, LAST_VALUE }

Public Member Functions

- EN_PassengerChoiceModel getModel () const
- std::string getModelAsString () const
- const std::string describe () const
- bool operator== (const EN_PassengerChoiceModel &) const
- PassengerChoiceModel (const EN_PassengerChoiceModel &)
- PassengerChoiceModel (const char iModel)
- PassengerChoiceModel (const PassengerChoiceModel &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PassengerChoiceModel &)
- static char getModelLabel (const EN_PassengerChoiceModel &)
- static std::string getModelLabelAsString (const EN_PassengerChoiceModel &)
- static std::string describeLabels ()

32.120.1 Detailed Description

Enumeration of passenger choice models.

Definition at line 15 of file PassengerChoiceModel.hpp.

32.120.2 Member Enumeration Documentation

32.120.2.1 enum stdair::PassengerChoiceModel::EN_PassengerChoiceModel

Enumerator:

HARD_RESTRICTION

PRICE_ORIENTED

HYBRID

LAST_VALUE

Definition at line 17 of file PassengerChoiceModel.hpp.

32.120.3 Constructor & Destructor Documentation

32.120.3.1 stdair::PassengerChoiceModel::PassengerChoiceModel (const EN_PassengerChoiceModel & *iPassengerChoiceModel*)

Constructor.

Definition at line 36 of file PassengerChoiceModel.cpp.

32.120.3.2 stdair::PassengerChoiceModel::PassengerChoiceModel (const char *iModel*)

Constructor.

Definition at line 41 of file PassengerChoiceModel.cpp.

References describeLabels(), HARD_RESTRICTION, HYBRID, LAST_VALUE, and PRICE_-ORIENTED.

32.120.3.3 stdair::PassengerChoiceModel::PassengerChoiceModel (const PassengerChoiceModel & *iPassengerChoiceModel*)

Default copy constructor.

Definition at line 30 of file PassengerChoiceModel.cpp.

32.120.4 Member Function Documentation

32.120.4.1 const std::string & stdair::PassengerChoiceModel::getLabel (const EN_PassengerChoiceModel & *iModel*) `[static]`

Get the label as a string (e.g., HardRestrictionModel", "PriceOrientedModel" or "HybridModel).

Definition at line 60 of file PassengerChoiceModel.cpp.

32.120.4.2 char stdair::PassengerChoiceModel::getModelLabel (const EN_PassengerChoiceModel & *iModel*) `[static]`

Get the label as a single char (e.g., 'R', 'P' or 'H').

Definition at line 65 of file PassengerChoiceModel.cpp.

32.120.4.3 std::string stdair::PassengerChoiceModel::getModelLabelAsString (const EN_PassengerChoiceModel & *iModel*) `[static]`

Get the label as a string of a single char (e.g., "R", "P" or "H").

Definition at line 71 of file PassengerChoiceModel.cpp.

32.120.4.4 std::string stdair::PassengerChoiceModel::describeLabels () `[static]`

List the labels.

Definition at line 78 of file PassengerChoiceModel.cpp.

References LAST_VALUE.

Referenced by PassengerChoiceModel().

32.120.4.5 PassengerChoiceModel::EN_PassengerChoiceModel stdair::PassengerChoiceModel::getModel () const

Get the enumerated value.

Definition at line 90 of file PassengerChoiceModel.cpp.

32.120.4.6 std::string stdair::PassengerChoiceModel::getModelAsString () const

Get the enumerated value as a short string (e.g., "R", "P" or "H").

Definition at line 95 of file PassengerChoiceModel.cpp.

32.120.4.7 const std::string stdair::PassengerChoiceModel::describe () const `[virtual]`

Give a description of the structure (e.g., HardRestrictionModel", "PriceOrientedModel" or "HybridModel).

Implements stdair::StructAbstract.

Definition at line 102 of file PassengerChoiceModel.cpp.

32.120.4.8 bool stdair::PassengerChoiceModel::operator== (const EN_PassengerChoiceModel & *iModel*) const

Comparaison operator.

Definition at line 110 of file PassengerChoiceModel.cpp.

32.120.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.120.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

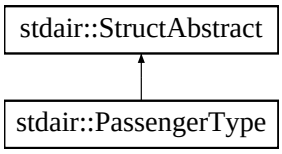
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/PassengerChoiceModel.hpp
- stdair/basic/PassengerChoiceModel.cpp

32.121 stdair::PassengerType Struct Reference

`#include <stdair/basic/PassengerType.hpp>`Inheritance diagram for stdair::PassengerType::



Public Types

- enum EN_PassengerType { LEISURE = 0, BUSINESS, FIRST, LAST_VALUE }

Public Member Functions

- EN_PassengerType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_PassengerType &) const
- PassengerType (const EN_PassengerType &)
- PassengerType (const char iType)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PassengerType &)
- static char getTypeLabel (const EN_PassengerType &)
- static std::string getTypeLabelAsString (const EN_PassengerType &)
- static std::string describeLabels ()

32.121.1 Detailed Description

Enumeration of Frequent Flyer types.

Definition at line 15 of file PassengerType.hpp.

32.121.2 Member Enumeration Documentation

32.121.2.1 enum stdair::PassengerType::EN_PassengerType

Enumerator:

LEISURE

BUSINESS

FIRST

LAST_VALUE

Definition at line 17 of file PassengerType.hpp.

32.121.3 Constructor & Destructor Documentation

32.121.3.1 stdair::PassengerType::PassengerType (const EN_PassengerType & *iPassengerType*)

Constructor.

Definition at line 21 of file PassengerType.cpp.

32.121.3.2 stdair::PassengerType::PassengerType (const char *iType*)

Constructor.

Definition at line 26 of file PassengerType.cpp.

References BUSINESS, describeLabels(), FIRST, LAST_VALUE, and LEISURE.

32.121.4 Member Function Documentation

32.121.4.1 const std::string & stdair::PassengerType::getLabel (const EN_PassengerType & *iType*) `[static]`

Get the label as a string (e.g., "Leisure" or "Business").

Definition at line 44 of file PassengerType.cpp.

32.121.4.2 char stdair::PassengerType::getTypeLabel (const EN_PassengerType & *iType*) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 49 of file PassengerType.cpp.

32.121.4.3 std::string stdair::PassengerType::getTypeLabelAsString (const EN_PassengerType & *iType*) [static]

Get the label as a single char (e.g., 'L' or 'B').

Definition at line 55 of file PassengerType.cpp.

32.121.4.4 std::string stdair::PassengerType::describeLabels () [static]

List the labels.

Definition at line 62 of file PassengerType.cpp.

References LAST_VALUE.

Referenced by PassengerType().

32.121.4.5 PassengerType::EN_PassengerType stdair::PassengerType::getType () const

Get the enumerated value.

Definition at line 74 of file PassengerType.cpp.

32.121.4.6 std::string stdair::PassengerType::getTypeAsString () const

Get the enumerated value as a short string (e.g., 'L' or 'B').

Definition at line 79 of file PassengerType.cpp.

32.121.4.7 const std::string stdair::PassengerType::describe () const [virtual]

Give a description of the structure (e.g., "Leisure" or "Business").

Implements stdair::StructAbstract.

Definition at line 86 of file PassengerType.cpp.

32.121.4.8 bool stdair::PassengerType::operator== (const EN_PassengerType & *iType*) const

Comparison operator.

Definition at line 93 of file PassengerType.cpp.

32.121.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.121.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

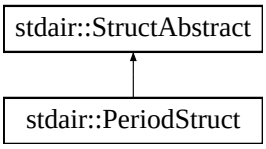
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/PassengerType.hpp
- stdair/basic/PassengerType.cpp

32.122 stdair::PeriodStruct Struct Reference

`#include <stdair/bom/PeriodStruct.hpp>`Inheritance diagram for stdair::PeriodStruct::



Public Member Functions

- const DatePeriod_T & getDateRange () const
- const DoWStruct & getDoW () const
- void setDateRange (const DatePeriod_T &iDateRange)

- void setDoW (const DoWStruct &iDoW)
- const std::string describe () const
- const std::string describeShort () const
- PeriodStruct addDateOffset (const DateOffset_T &) const
- PeriodStruct intersection (const PeriodStruct &) const
- const bool isValid () const
- PeriodStruct (const DatePeriod_T &, const DoWStruct &)
- PeriodStruct ()
- PeriodStruct (const PeriodStruct &)
- ~PeriodStruct ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

32.122.1 Detailed Description

Define a departure period

A period is defined by a date range and a day-of-week struct.

Definition at line 19 of file PeriodStruct.hpp.

32.122.2 Constructor & Destructor Documentation

32.122.2.1 stdair::PeriodStruct::PeriodStruct (const DatePeriod_T & *iDateRange*, const DoWStruct & *iDoW*)

Constructor.

Definition at line 19 of file PeriodStruct.cpp.

32.122.2.2 stdair::PeriodStruct::PeriodStruct ()

Default constructors.

Definition at line 14 of file PeriodStruct.cpp.

Referenced by addDateOffset(), and intersection().

32.122.2.3 stdair::PeriodStruct::PeriodStruct (const PeriodStruct & *iPeriodStruct*)

Definition at line 25 of file PeriodStruct.cpp.

32.122.2.4 stdair::PeriodStruct::~PeriodStruct () `[inline]`

Default destructor.

Definition at line 64 of file PeriodStruct.hpp.

32.122.3 Member Function Documentation

32.122.3.1 const DatePeriod_T& stdair::PeriodStruct::getDateRange () const [inline]

Retrieve the attributes.

Definition at line 23 of file PeriodStruct.hpp.

Referenced by addDateOffset().

32.122.3.2 const DoWStruct& stdair::PeriodStruct::getDoW () const [inline]

Definition at line 26 of file PeriodStruct.hpp.

Referenced by addDateOffset().

32.122.3.3 void stdair::PeriodStruct::setDateRange (const DatePeriod_T & *iDateRange*) [inline]

Set the new value for the attributes.

Definition at line 33 of file PeriodStruct.hpp.

32.122.3.4 void stdair::PeriodStruct::setDoW (const DoWStruct & *iDoW*) [inline]

Definition at line 36 of file PeriodStruct.hpp.

32.122.3.5 const std::string stdair::PeriodStruct::describe () const [virtual]

Display explicitly (e.g., "Mon.Tue.Wed.Thu.Fri.").

Implements stdair::StructAbstract.

Definition at line 38 of file PeriodStruct.cpp.

References stdair::DoWStruct::describe().

32.122.3.6 const std::string stdair::PeriodStruct::describeShort () const

Display as a bit set (e.g., "1111100").

Definition at line 31 of file PeriodStruct.cpp.

References stdair::DoWStruct::describeShort().

Referenced by stdair::FlightPeriodKey::toString().

32.122.3.7 PeriodStruct stdair::PeriodStruct::addDateOffset (const DateOffset_T & *iDateOffset*) const

Build a period struct from this period struct by adding a date offset.

Definition at line 46 of file PeriodStruct.cpp.

References getDateRange(), getDoW(), PeriodStruct(), and stdair::DoWStruct::shift().

32.122.3.8 PeriodStruct stdair::PeriodStruct::intersection (const PeriodStruct & *iPeriodStruct*) const

Build a new period struct which is the intersection of two period structs.

Definition at line 63 of file PeriodStruct.cpp.

References stdair::DoWStruct::intersection(), and PeriodStruct().

32.122.3.9 const bool stdair::PeriodStruct::isValid () const

Return if the period is valid (i.e., valid date range and valid DoW).

Definition at line 72 of file PeriodStruct.cpp.

References stdair::DoWStruct::isValid().

32.122.3.10 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.122.3.11 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

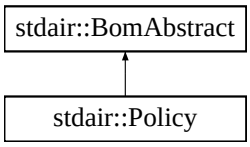
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/bom/PeriodStruct.hpp
- stdair/bom/PeriodStruct.cpp

32.123 stdair::Policy Class Reference

`#include <stdair/bom/Policy.hpp>`Inheritance diagram for stdair::Policy::



Public Types

- typedef PolicyKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const BookingClassList_T & getBookingClassList () const
- const NbOfBookings_T & getDemand () const
- const StdDevValue_T & getStdDev () const
- const Yield_T & getYield () const
- const Revenue_T getTotalRevenue () const
- void setDemand (const NbOfBookings_T &iDemand)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setYield (const Yield_T &iYield)
- void resetDemandForecast ()
- void addYieldDemand (const Yield_T &, const NbOfBookings_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- Policy (const Key_T &)
- virtual ~Policy ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

32.123.1 Detailed Description

Structure holding the elements of a policy. A policy is a set of booking classes, each booking class belongs to a different Fare Family.

Definition at line 30 of file Policy.hpp.

32.123.2 Member Typedef Documentation

32.123.2.1 typedef PolicyKey stdair::Policy::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 40 of file Policy.hpp.

32.123.3 Constructor & Destructor Documentation

32.123.3.1 stdair::Policy::Policy (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 31 of file Policy.cpp.

32.123.3.2 stdair::Policy::~Policy () `[protected, virtual]`

Destructor.

Definition at line 35 of file Policy.cpp.

32.123.4 Member Function Documentation

32.123.4.1 const Key_T& stdair::Policy::getKey () const `[inline]`

Get the policy key.

Definition at line 45 of file Policy.hpp.

32.123.4.2 BomAbstract* const stdair::Policy::getParent () const `[inline]`

Get the parent object.

Definition at line 50 of file Policy.hpp.

32.123.4.3 const HolderMap_T& stdair::Policy::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 57 of file Policy.hpp.

32.123.4.4 const BookingClassList_T & stdair::Policy::getBookingClassList () const

Getter for the booking classes.

Definition at line 52 of file Policy.cpp.

32.123.4.5 const NbOfBookings_T& stdair::Policy::getDemand () const [inline]

Getter for the demand.

Definition at line 65 of file Policy.hpp.

32.123.4.6 const StdDevValue_T& stdair::Policy::getStdDev () const [inline]

Getter for the standard deviation demand.

Definition at line 70 of file Policy.hpp.

32.123.4.7 const Yield_T& stdair::Policy::getYield () const [inline]

Getter for the yield.

Definition at line 75 of file Policy.hpp.

32.123.4.8 const Revenue_T stdair::Policy::getTotalRevenue () const

Get the total revenue of the policy.

Definition at line 57 of file Policy.cpp.

32.123.4.9 void stdair::Policy::setDemand (const NbOfBookings_T & *iDemand*) [inline]

Setter for the unconstraining demand.

Definition at line 85 of file Policy.hpp.

32.123.4.10 void stdair::Policy::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Setter for standard deviation demand.

Definition at line 90 of file Policy.hpp.

32.123.4.11 void stdair::Policy::setYield (const Yield_T & *iYield*) [inline]

Setter for the yield.

Definition at line 95 of file Policy.hpp.

32.123.4.12 void stdair::Policy::resetDemandForecast () [inline]

Reset demand forecast.

Definition at line 100 of file Policy.hpp.

32.123.4.13 void stdair::Policy::addYieldDemand (const Yield_T & *iYield*, const NbOfBookings_T & *iDemand*)

Add the new pair (yield, demand) to the map.

Definition at line 70 of file Policy.cpp.

32.123.4.14 void stdair::Policy::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 116 of file Policy.hpp.

References toString().

32.123.4.15 void stdair::Policy::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 125 of file Policy.hpp.

32.123.4.16 std::string stdair::Policy::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 39 of file Policy.cpp.

References describeKey().

Referenced by stdair::SegmentCabin::describeConvexHull(), and toStream().

32.123.4.17 const std::string stdair::Policy::describeKey () const [inline]

Get a string describing the key.

Definition at line 136 of file Policy.hpp.

References stdair::PolicyKey::toString().

Referenced by toString().

32.123.4.18 template<class Archive > void stdair::Policy::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

32.123.5 Friends And Related Function Documentation

32.123.5.1 friend class FacBom [friend]

Definition at line 31 of file Policy.hpp.

32.123.5.2 friend class FacBomManager [friend]

Definition at line 32 of file Policy.hpp.

32.123.5.3 friend class boost::serialization::access [friend]

Definition at line 33 of file Policy.hpp.

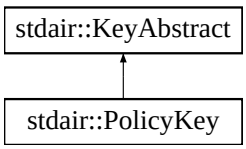
The documentation for this class was generated from the following files:

- stdair/bom/Policy.hpp
- stdair/bom/Policy.cpp

32.124 stdair::PolicyKey Struct Reference

Key of a given policy, made of a policy code.

`#include <stdair/bom/PolicyKey.hpp>`Inheritance diagram for stdair::PolicyKey::



Public Member Functions

- PolicyKey (const PolicyCode_T &iPolicyCode)
- PolicyKey (const PolicyKey &)
- ~PolicyKey ()

- const PolicyCode_T & getPolicyCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.124.1 Detailed Description

Key of a given policy, made of a policy code.

Definition at line 26 of file PolicyKey.hpp.

32.124.2 Constructor & Destructor Documentation

32.124.2.1 stdair::PolicyKey::PolicyKey (const PolicyCode_T & *iPolicyCode*)

Constructor.

Definition at line 28 of file PolicyKey.cpp.

32.124.2.2 stdair::PolicyKey::PolicyKey (const PolicyKey & *iPolicyKey*)

Copy constructor.

Definition at line 23 of file PolicyKey.cpp.

32.124.2.3 stdair::PolicyKey::~PolicyKey ()

Destructor.

Definition at line 33 of file PolicyKey.cpp.

32.124.3 Member Function Documentation

32.124.3.1 const PolicyCode_T& stdair::PolicyKey::getPolicyCode () const `[inline]`

Get the policy code.

Definition at line 56 of file PolicyKey.hpp.

32.124.3.2 void stdair::PolicyKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file PolicyKey.cpp.

References toString().

32.124.3.3 void stdair::PolicyKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file PolicyKey.cpp.

32.124.3.4 const std::string stdair::PolicyKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file PolicyKey.cpp.

Referenced by stdair::Policy::describeKey(), and toStream().

32.124.3.5 template<class Archive > void stdair::PolicyKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 68 of file PolicyKey.cpp.

32.124.4 Friends And Related Function Documentation

32.124.4.1 friend class boost::serialization::access [friend]

Definition at line 27 of file PolicyKey.hpp.

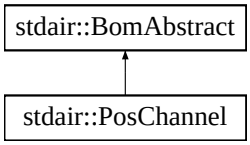
The documentation for this struct was generated from the following files:

- stdair/bom/PolicyKey.hpp
- stdair/bom/PolicyKey.cpp

32.125 stdair::PosChannel Class Reference

Class representing the actual attributes for a fare point of sale.

`#include <stdair/bom/PosChannel.hpp>`Inheritance diagram for stdair::PosChannel::



Public Types

- typedef PosChannelKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const stdair::HolderMap_T & getHolderMap () const
- const CityCode_T & getPos () const
- const ChannelLabel_T & getChannel () const

Protected Member Functions

- PosChannel (const Key_T &)
- virtual ~PosChannel ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.125.1 Detailed Description

Class representing the actual attributes for a fare point of sale.

Definition at line 19 of file PosChannel.hpp.

32.125.2 Member Typedef Documentation

32.125.2.1 typedef PosChannelKey stdair::PosChannel::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file PosChannel.hpp.

32.125.3 Constructor & Destructor Documentation

32.125.3.1 stdair::PosChannel::PosChannel (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 28 of file PosChannel.cpp.

32.125.3.2 stdair::PosChannel::~PosChannel () `[protected, virtual]`

Destructor.

Definition at line 33 of file PosChannel.cpp.

32.125.4 Member Function Documentation

32.125.4.1 void stdair::PosChannel::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file PosChannel.hpp.

References toString().

32.125.4.2 void stdair::PosChannel::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file PosChannel.hpp.

32.125.4.3 std::string stdair::PosChannel::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file PosChannel.cpp.

References describeKey().

Referenced by toStream().

32.125.4.4 const std::string stdair::PosChannel::describeKey () const [inline]

Get a string describing the key.

Definition at line 58 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::toString().

Referenced by toString().

32.125.4.5 const Key_T& stdair::PosChannel::getKey () const [inline]

Get the primary key (pos, channel).

Definition at line 67 of file PosChannel.hpp.

References _key.

32.125.4.6 BomAbstract* const stdair::PosChannel::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file PosChannel.hpp.

References _parent.

32.125.4.7 const stdair::HolderMap_T& stdair::PosChannel::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file PosChannel.hpp.

References _holderMap.

32.125.4.8 const CityCode_T& stdair::PosChannel::getPos () const [inline]

Get the point-of-sale.

Definition at line 88 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::getPos().

32.125.4.9 const ChannelLabel_T& stdair::PosChannel::getChannel () const [inline]

Get the channel.

Definition at line 95 of file PosChannel.hpp.

References _key, and stdair::PosChannelKey::getChannel().

32.125.5 Friends And Related Function Documentation

32.125.5.1 friend class FacBom [friend]

Definition at line 20 of file PosChannel.hpp.

32.125.5.2 friend class FacCloneBom [friend]

Definition at line 21 of file PosChannel.hpp.

32.125.5.3 friend class FacBomManager [friend]

Definition at line 22 of file PosChannel.hpp.

32.125.6 Member Data Documentation

32.125.6.1 Key_T stdair::PosChannel::_key [protected]

Primary key (flight number and departure date).

Definition at line 127 of file PosChannel.hpp.

Referenced by describeKey(), getChannel(), getKey(), and getPos().

32.125.6.2 BomAbstract* stdair::PosChannel::_parent [protected]

Pointer on the parent class.

Definition at line 132 of file PosChannel.hpp.

Referenced by getParent().

32.125.6.3 HolderMap_T stdair::PosChannel::_holderMap [protected]

Map holding the children.

Definition at line 137 of file PosChannel.hpp.

Referenced by getHolderMap().

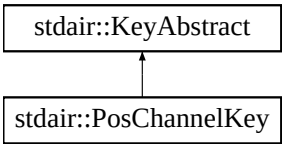
The documentation for this class was generated from the following files:

- stdair/bom/PosChannel.hpp
- stdair/bom/PosChannel.cpp

32.126 stdair::PosChannelKey Struct Reference

Key of point of sale and channel.

`#include <stdair/bom/PosChannelKey.hpp>`Inheritance diagram for stdair::PosChannelKey::



Public Member Functions

- PosChannelKey (const stdair::CityCode_T &, const stdair::ChannelLabel_T &)
- PosChannelKey (const PosChannelKey &)
- ~PosChannelKey ()
- const stdair::CityCode_T & getPos () const
- const stdair::ChannelLabel_T & getChannel () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.126.1 Detailed Description

Key of point of sale and channel.

Definition at line 15 of file PosChannelKey.hpp.

32.126.2 Constructor & Destructor Documentation

32.126.2.1 stdair::PosChannelKey::PosChannelKey (const stdair::CityCode_T & *iPos*, const stdair::ChannelLabel_T & *iChannel*)

Main constructor.

Definition at line 22 of file PosChannelKey.cpp.

32.126.2.2 stdair::PosChannelKey::PosChannelKey (const PosChannelKey & *iKey*)

Copy constructor.

Definition at line 28 of file PosChannelKey.cpp.

32.126.2.3 stdair::PosChannelKey::~PosChannelKey ()

Destructor.

Definition at line 33 of file PosChannelKey.cpp.

32.126.3 Member Function Documentation

32.126.3.1 const stdair::CityCode_T& stdair::PosChannelKey::getPos () const [inline]

Get the point of sale.

Definition at line 43 of file PosChannelKey.hpp.

Referenced by stdair::PosChannel::getPos().

32.126.3.2 const stdair::ChannelLabel_T& stdair::PosChannelKey::getChannel () const [inline]

Get the channel.

Definition at line 50 of file PosChannelKey.hpp.

Referenced by stdair::PosChannel::getChannel().

32.126.3.3 void stdair::PosChannelKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file PosChannelKey.cpp.

References toString().

32.126.3.4 void stdair::PosChannelKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file PosChannelKey.cpp.

32.126.3.5 const std::string stdair::PosChannelKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file PosChannelKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

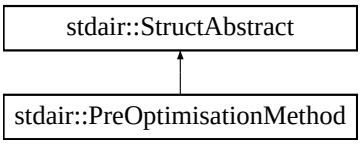
Referenced by stdair::PosChannel::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/PosChannelKey.hpp
- stdair/bom/PosChannelKey.cpp

32.127 stdair::PreOptimisationMethod Struct Reference

`#include <stdair/basic/PreOptimisationMethod.hpp>`Inheritance diagram for stdair::PreOptimisationMethod::



Public Types

- enum EN_PreOptimisationMethod { NONE = 0, FA, MRT, LAST_VALUE }

Public Member Functions

- EN_PreOptimisationMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_PreOptimisationMethod &) const
- PreOptimisationMethod (const EN_PreOptimisationMethod &)
- PreOptimisationMethod (const char iMethod)
- PreOptimisationMethod (const PreOptimisationMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_PreOptimisationMethod &)
- static char getMethodLabel (const EN_PreOptimisationMethod &)
- static std::string getMethodLabelAsString (const EN_PreOptimisationMethod &)
- static std::string describeLabels ()

32.127.1 Detailed Description

Enumeration of PreOptimisation methods.

Definition at line 15 of file PreOptimisationMethod.hpp.

32.127.2 Member Enumeration Documentation

32.127.2.1 enum stdair::PreOptimisationMethod::EN_PreOptimisationMethod

Enumerator:

NONE

FA

MRT

LAST_VALUE

Definition at line 17 of file PreOptimisationMethod.hpp.

32.127.3 Constructor & Destructor Documentation

32.127.3.1 stdair::PreOptimisationMethod::PreOptimisationMethod (const EN_PreOptimisationMethod & *iPreOptimisationMethod*)

Constructor.

Definition at line 36 of file PreOptimisationMethod.cpp.

32.127.3.2 stdair::PreOptimisationMethod::PreOptimisationMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file PreOptimisationMethod.cpp.

References describeLabels(), FA, LAST_VALUE, MRT, and NONE.

32.127.3.3 stdair::PreOptimisationMethod::PreOptimisationMethod (const PreOptimisationMethod & *iPreOptimisationMethod*)

Default copy constructor.

Definition at line 30 of file PreOptimisationMethod.cpp.

32.127.4 Member Function Documentation

32.127.4.1 const std::string & stdair::PreOptimisationMethod::getLabel (const EN_PreOptimisationMethod & *iMethod*) `[static]`

Get the label as a string (e.g., MRT or FA).

Definition at line 60 of file PreOptimisationMethod.cpp.

32.127.4.2 char stdair::PreOptimisationMethod::getMethodLabel (const EN_PreOptimisationMethod & *iMethod*) `[static]`

Get the label as a single char (e.g., 'M' or 'E').

Definition at line 65 of file PreOptimisationMethod.cpp.

32.127.4.3 std::string stdair::PreOptimisationMethod::getMethodLabelAsString (const EN_PreOptimisationMethod & *iMethod*) `[static]`

Get the label as a string of a single char (e.g., "M" or "E").

Definition at line 71 of file PreOptimisationMethod.cpp.

32.127.4.4 std::string stdair::PreOptimisationMethod::describeLabels () [static]

List the labels.

Definition at line 78 of file PreOptimisationMethod.cpp.

References LAST_VALUE.

Referenced by PreOptimisationMethod().

32.127.4.5 PreOptimisationMethod::EN_PreOptimisationMethod stdair::PreOptimisationMethod::getMethod () const

Get the enumerated value.

Definition at line 90 of file PreOptimisationMethod.cpp.

Referenced by stdair::AirlineFeature::getPreOptimisationMethod().

32.127.4.6 std::string stdair::PreOptimisationMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "M" or "E").

Definition at line 95 of file PreOptimisationMethod.cpp.

32.127.4.7 const std::string stdair::PreOptimisationMethod::describe () const [virtual]

Give a description of the structure (e.g., MRT or FA).

Implements stdair::StructAbstract.

Definition at line 102 of file PreOptimisationMethod.cpp.

32.127.4.8 bool stdair::PreOptimisationMethod::operator== (const EN_PreOptimisationMethod & *iMethod*) const

Comparaison operator.

Definition at line 110 of file PreOptimisationMethod.cpp.

32.127.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct,

stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.127.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

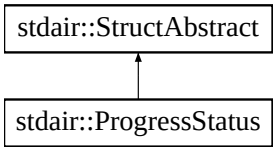
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/PreOptimisationMethod.hpp
- stdair/basic/PreOptimisationMethod.cpp

32.128 stdair::ProgressStatus Struct Reference

`#include <stdair/basic/ProgressStatus.hpp>`Inheritance diagram for stdair::ProgressStatus::



Public Member Functions

- const Count_T & count () const
- const Count_T & getCurrentNb () const
- const Count_T & getExpectedNb () const
- const Count_T & getActualNb () const
- const ProgressPercentage_T progress () const
- void setCurrentNb (const Count_T &iCurrentNb)
- void setExpectedNb (const Count_T &iExpectedNb)
- void setActualNb (const Count_T &iActualNb)

- void reset ()
- Count_T operator+= (Count_T iIncrement)
- Count_T operator++ ()
- const std::string describe () const
- const std::string toString () const
- ProgressStatus (const Count_T &iCurrentNb, const Count_T &iExpectedNb, const Count_T &iActualNb)
- ProgressStatus (const Count_T &iExpectedNb, const Count_T &iActualNb)
- ProgressStatus (const Count_T &iActualNb)
- ProgressStatus ()
- ProgressStatus (const ProgressStatus &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

32.128.1 Detailed Description

Structure holding the details of a progress status.

The progress status is given by the ratio between the "current" and the "expected" (or "actual") numbers. For instance, when the expected/actual number is 1000 and the current number is 200, then the progress status is 20% (= 200 / 1000).

Definition at line 27 of file ProgressStatus.hpp.

32.128.2 Constructor & Destructor Documentation

32.128.2.1 stdair::ProgressStatus::ProgressStatus (const Count_T & *iCurrentNb*, const Count_T & *iExpectedNb*, const Count_T & *iActualNb*)

Constructor.

Parameters:

const Count_T& The current number.

const Count_T& The expected number.

const Count_T& The actual number.

Definition at line 15 of file ProgressStatus.cpp.

32.128.2.2 stdair::ProgressStatus::ProgressStatus (const Count_T & *iExpectedNb*, const Count_T & *iActualNb*)

Constructor.

As no current number is given, it is set to 0.

Parameters:

const Count_T& The expected number.

const Count_T& The actual number.

Definition at line 23 of file ProgressStatus.cpp.

32.128.2.3 stdair::ProgressStatus::ProgressStatus (const Count_T & *iActualNb*)

Constructor.

As no expected number is given, it is assumed to be equal to the actual one. The current number is set to 0.

Parameters:

const Count_T& The actual number.

Definition at line 30 of file ProgressStatus.cpp.

32.128.2.4 stdair::ProgressStatus::ProgressStatus ()

Constructor.

All the numbers are set to 0.

Definition at line 36 of file ProgressStatus.cpp.

32.128.2.5 stdair::ProgressStatus::ProgressStatus (const ProgressStatus & *iProgressStatus*)

Copy Constructor.

Definition at line 43 of file ProgressStatus.cpp.

32.128.3 Member Function Documentation

32.128.3.1 const Count_T& stdair::ProgressStatus::count () const `[inline]`

Get the current number.

Definition at line 31 of file ProgressStatus.hpp.

32.128.3.2 const Count_T& stdair::ProgressStatus::getCurrentNb () const `[inline]`

Get the current number.

Definition at line 36 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

32.128.3.3 const Count_T& stdair::ProgressStatus::getExpectedNb () const `[inline]`

Get the expected number.

Definition at line 41 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

32.128.3.4 const Count_T& stdair::ProgressStatus::getActualNb () const `[inline]`

Get the actual number.

Definition at line 46 of file ProgressStatus.hpp.

Referenced by stdair::ProgressStatusSet::describe().

32.128.3.5 const ProgressPercentage_T stdair::ProgressStatus::progress () const [inline]

Get the progress as a percentage.

Definition at line 51 of file ProgressStatus.hpp.

References stdair::MAXIMUM_PROGRESS_STATUS.

Referenced by toString().

32.128.3.6 void stdair::ProgressStatus::setCurrentNb (const Count_T & *iCurrentNb*) [inline]

Set the current number.

Definition at line 65 of file ProgressStatus.hpp.

32.128.3.7 void stdair::ProgressStatus::setExpectedNb (const Count_T & *iExpectedNb*) [inline]

Set the expected number.

Definition at line 70 of file ProgressStatus.hpp.

32.128.3.8 void stdair::ProgressStatus::setActualNb (const Count_T & *iActualNb*) [inline]

Set the actual number.

Definition at line 75 of file ProgressStatus.hpp.

32.128.3.9 void stdair::ProgressStatus::reset ()

Reset the current number (to 0).

Definition at line 50 of file ProgressStatus.cpp.

References stdair::DEFAULT_PROGRESS_STATUS.

32.128.3.10 Count_T stdair::ProgressStatus::operator+= (Count_T *iIncrement*) [inline]

Increment the current number.

Definition at line 83 of file ProgressStatus.hpp.

32.128.3.11 Count_T stdair::ProgressStatus::operator++ () [inline]

Increment the current number.

Definition at line 89 of file ProgressStatus.hpp.

32.128.3.12 const std::string stdair::ProgressStatus::describe () const [virtual]

Give a description of the structure (e.g., "1 {99, 100}").

Implements stdair::StructAbstract.

Definition at line 56 of file ProgressStatus.cpp.

32.128.3.13 const std::string stdair::ProgressStatus::toString () const

Give a description of the structure (e.g., "1% (1/ 100)").

Definition at line 63 of file ProgressStatus.cpp.

References progress().

32.128.3.14 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.128.3.15 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

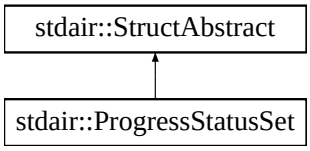
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/ProgressStatus.hpp
- stdair/basic/ProgressStatus.cpp

32.129 stdair::ProgressStatusSet Struct Reference

`#include <stdair/basic/ProgressStatusSet.hpp>`Inheritance diagram for stdair::ProgressStatusSet::



Public Member Functions

- const ProgressStatus & getTypeSpecificStatus () const
- const ProgressStatus & getSpecificGeneratorStatus () const
- const ProgressStatus & getOverallStatus () const
- void setTypeSpecificStatus (const ProgressStatus &iProgressStatus)
- void setSpecificGeneratorStatus (const ProgressStatus &iProgressStatus, const EventGeneratorKey_T &iKey)
- void setOverallStatus (const ProgressStatus &iProgressStatus)
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- ProgressStatusSet (const EventType::EN_EventType &)
- ProgressStatusSet (const ProgressStatusSet &)
- ~ProgressStatusSet ()
- void toStream (std::ostream &ioOut) const

32.129.1 Detailed Description

Structrure holding a set of progress status.

Definition at line 22 of file ProgressStatusSet.hpp.

32.129.2 Constructor & Destructor Documentation

32.129.2.1 stdair::ProgressStatusSet::ProgressStatusSet (const EventType::EN_EventType & *iType*)

Constructor .

Definition at line 20 of file ProgressStatusSet.cpp.

32.129.2.2 stdair::ProgressStatusSet::ProgressStatusSet (const ProgressStatusSet & *iProgressStatusSet*)

Copy constructor.

Definition at line 27 of file ProgressStatusSet.cpp.

32.129.2.3 stdair::ProgressStatusSet::~ProgressStatusSet ()

Destructor.

Definition at line 36 of file ProgressStatusSet.cpp.

32.129.3 Member Function Documentation

32.129.3.1 const ProgressStatus& stdair::ProgressStatusSet::getTypeSpecificStatus () const [inline]

Get the progress status specific to that event type.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 31 of file ProgressStatusSet.hpp.

32.129.3.2 const ProgressStatus& stdair::ProgressStatusSet::getSpecificGeneratorStatus () const [inline]

Get the progress status specific to the content key for that event.

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 43 of file ProgressStatusSet.hpp.

32.129.3.3 const ProgressStatus& stdair::ProgressStatusSet::getOverallStatus () const [inline]

Get the overall progress status (absolute, for all the events).

Note that that progress status may not be up-to-date. That attribute is up-to-date only after a call to the popEvent() method of SEvMgr.

Definition at line 54 of file ProgressStatusSet.hpp.

32.129.3.4 void stdair::ProgressStatusSet::setTypeSpecificStatus (const ProgressStatus & *iProgressStatus*) [inline]

Set/update the progress status specific to that event type.

Definition at line 62 of file ProgressStatusSet.hpp.

32.129.3.5 void stdair::ProgressStatusSet::setSpecificGeneratorStatus (const ProgressStatus & *iProgressStatus*, const EventGeneratorKey_T & *iKey*) [inline]

Set/update the progress status specific to the content key for that event.

Definition at line 68 of file ProgressStatusSet.hpp.

32.129.3.6 void stdair::ProgressStatusSet::setOverallStatus (const ProgressStatus & *iProgressStatus*) [inline]

Set/update the overall progress status (absolute, for all the events).

Definition at line 76 of file ProgressStatusSet.hpp.

32.129.3.7 void stdair::ProgressStatusSet::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 40 of file ProgressStatusSet.cpp.

32.129.3.8 const std::string stdair::ProgressStatusSet::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 44 of file ProgressStatusSet.cpp.

References stdair::ProgressStatus::getActualNb(), stdair::ProgressStatus::getCurrentNb(), and stdair::ProgressStatus::getExpectedNb().

32.129.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

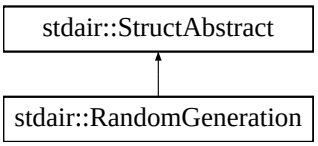
The documentation for this struct was generated from the following files:

- stdair/basic/ProgressStatusSet.hpp
- stdair/basic/ProgressStatusSet.cpp

32.130 stdair::RandomGeneration Struct Reference

Class holding a random generator.

`#include <stdair/basic/RandomGeneration.hpp>`Inheritance diagram for stdair::RandomGeneration::



Public Member Functions

- RealNumber_T generateUniform01 ()
- RealNumber_T operator() ()
- RealNumber_T generateUniform (const RealNumber_T &, const RealNumber_T &)
- RealNumber_T generateNormal (const RealNumber_T &, const RealNumber_T &)
- RealNumber_T generateExponential (const RealNumber_T &)
- BaseGenerator_T & getBaseGenerator ()
- const std::string describe () const
- RandomGeneration (const RandomSeed_T &)
- RandomGeneration ()
- ~RandomGeneration ()
- void init (const RandomSeed_T &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Public Attributes

- BaseGenerator_T _generator

32.130.1 Detailed Description

Class holding a random generator.

Definition at line 17 of file RandomGeneration.hpp.

32.130.2 Constructor & Destructor Documentation

32.130.2.1 stdair::RandomGeneration::RandomGeneration (const RandomSeed_T & *iSeed*)

Main constructor.

Definition at line 27 of file RandomGeneration.cpp.

32.130.2.2 stdair::RandomGeneration::RandomGeneration ()

Default constructor.

Note:

As per Boost bug #3516 (https://svn.boost.org/trac/boost/ticket/3516) the seed should not be set to 0 (at least on versions of Boost lower than 1.44).

Definition at line 23 of file RandomGeneration.cpp.

32.130.2.3 stdair::RandomGeneration::~RandomGeneration ()

Destructor.

Definition at line 37 of file RandomGeneration.cpp.

32.130.3 Member Function Documentation

32.130.3.1 RealNumber_T stdair::RandomGeneration::generateUniform01 ()

Generate a randomised number following a uniform distribution between 0 (included) and 1 (excluded).

Definition at line 53 of file RandomGeneration.cpp.

References _generator.

Referenced by generateNormal(), generateUniform(), and operator()().

32.130.3.2 RealNumber_T stdair::RandomGeneration::operator() () [inline]

Same as generateUniform01(). That operator is provided for convenient reasons.

Definition at line 30 of file RandomGeneration.hpp.

References generateUniform01().

32.130.3.3 RealNumber_T stdair::RandomGeneration::generateUniform (const RealNumber_T & *iMinValue*, const RealNumber_T & *iMaxValue*)

Generate a randomized number following a uniform distribution between a minimum (included) and a maximum (excluded) value.

Definition at line 59 of file RandomGeneration.cpp.

References generateUniform01().

32.130.3.4 RealNumber_T stdair::RandomGeneration::generateNormal (const RealNumber_T & *mu*, const RealNumber_T & *sigma*)

Generate a randomized number following a normal distribution specified by a mean and a standard deviation.

Definition at line 68 of file RandomGeneration.cpp.

References generateUniform01().

Referenced by stdair::BookingClass::generateDemandSamples().

32.130.3.5 RealNumber_T stdair::RandomGeneration::generateExponential (const RealNumber_T & *lambda*)

Generate a randomized number following an exponential distribution specified by a mean and a lambda parameter.

Definition at line 86 of file RandomGeneration.cpp.

References _generator.

32.130.3.6 BaseGenerator_T& stdair::RandomGeneration::getBaseGenerator () `[inline]`

Retrieve the base generator for initialising other random generators.

Definition at line 56 of file RandomGeneration.hpp.

References _generator.

32.130.3.7 const std::string stdair::RandomGeneration::describe () const `[virtual]`

Give a description of the structure (for display purposes).

Implements stdair::StructAbstract.

Definition at line 46 of file RandomGeneration.cpp.

References _generator.

32.130.3.8 void stdair::RandomGeneration::init (const RandomSeed_T & *iSeed*)

Initialise the random generator.

A uniform random number distribution is defined, which produces "real" values between 0 and 1 (0 inclusive, 1 exclusive).

Definition at line 41 of file RandomGeneration.cpp.

References _generator.

32.130.3.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.130.3.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

32.130.4 Member Data Documentation

32.130.4.1 BaseGenerator_T stdair::RandomGeneration::_generator

Random number generator engine.

The random number generator is currently based on boost::minstd_rand. Alternates are boost::mt19937, boost::ecuyer1988.

Definition at line 112 of file RandomGeneration.hpp.

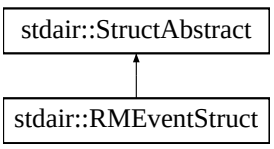
Referenced by describe(), generateExponential(), generateUniform01(), getBaseGenerator(), and init().

The documentation for this struct was generated from the following files:

- stdair/basic/RandomGeneration.hpp
- stdair/basic/RandomGeneration.cpp

32.131 stdair::RMEventStruct Struct Reference

`#include <stdair/bom/RMEventStruct.hpp>`Inheritance diagram for stdair::RMEventStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const KeyDescription_T & getFlightDateDescription () const
- const DateTime_T & getRMEventTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- RMEventStruct (const AirlineCode_T &, const KeyDescription_T &, const DateTime_T &)
- RMEventStruct (const RMEventStruct &)
- RMEventStruct ()
- ~RMEventStruct ()

32.131.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file RMEventStruct.hpp.

32.131.2 Constructor & Destructor Documentation

32.131.2.1 stdair::RMEventStruct::RMEventStruct (const AirlineCode_T & *iAirlineCode*, const KeyDescription_T & *iFlightDateDescription*, const DateTime_T & *iRMEventTime*)

Constructor.

Definition at line 27 of file RMEventStruct.cpp.

32.131.2.2 stdair::RMEventStruct::RMEventStruct (const RMEventStruct & *iRMEvent*)

Copy constructor.

Definition at line 19 of file RMEventStruct.cpp.

32.131.2.3 stdair::RMEventStruct::RMEventStruct ()

Default constructor.

It is private so that it can not be used.

Definition at line 13 of file RMEventStruct.cpp.

32.131.2.4 stdair::RMEventStruct::~RMEventStruct ()

Destructor.

Definition at line 36 of file RMEventStruct.cpp.

32.131.3 Member Function Documentation

32.131.3.1 const AirlineCode_T& stdair::RMEventStruct::getAirlineCode () const [inline]

Get the airline code.

Definition at line 23 of file RMEventStruct.hpp.

32.131.3.2 const KeyDescription_T& stdair::RMEventStruct::getFlightDateDescription () const [inline]

Get the string describing the flight-date key.

Definition at line 28 of file RMEventStruct.hpp.

32.131.3.3 const DateTime_T& stdair::RMEventStruct::getRMEventTime () const [inline]

Get the snapshot action time.

Definition at line 33 of file RMEventStruct.hpp.

32.131.3.4 void stdair::RMEventStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 40 of file RMEventStruct.cpp.

References describe().

32.131.3.5 void stdair::RMEventStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 45 of file RMEventStruct.cpp.

32.131.3.6 const std::string stdair::RMEventStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 49 of file RMEventStruct.cpp.

Referenced by toStream().

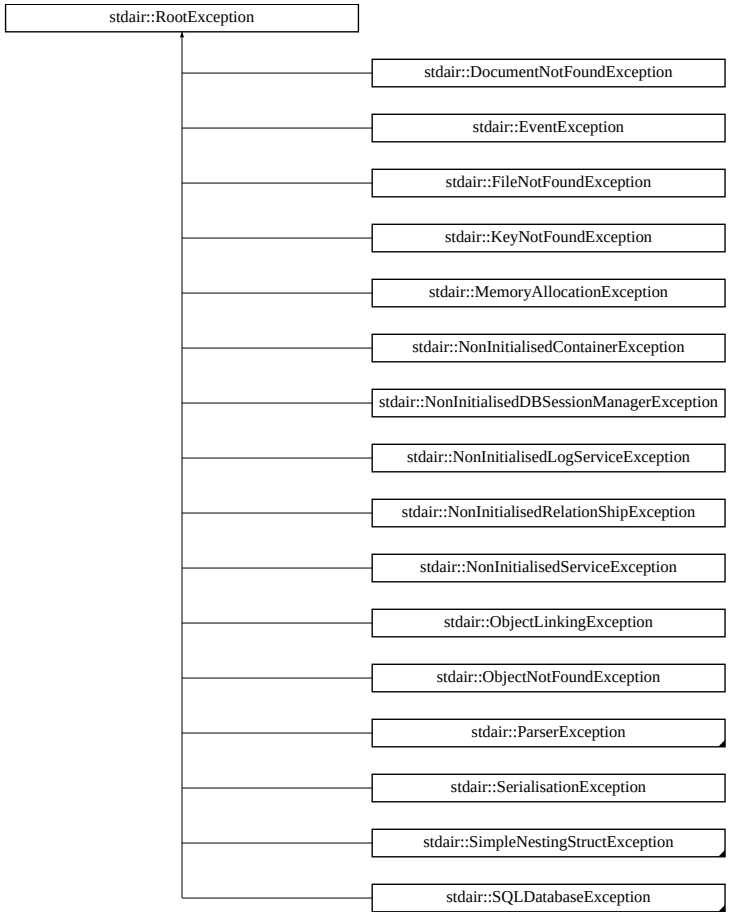
The documentation for this struct was generated from the following files:

- stdair/bom/RMEventStruct.hpp
- stdair/bom/RMEventStruct.cpp

32.132 stdair::RootException Class Reference

Root of the stdair exceptions.

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::RootException::



Public Member Functions

- RootException (const std::string &iWhat)
- RootException ()
- virtual ~RootException () throw ()
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.132.1 Detailed Description

Root of the stdair exceptions. All the stdair exceptions inherit from that root, allowing to catch them and to spot them easily when arising in code wrapping the stdair library.

Definition at line 19 of file stdair_exceptions.hpp.

32.132.2 Constructor & Destructor Documentation

32.132.2.1 stdair::RootException::RootException (const std::string & *iWhat*) [inline]

Main Constructor.

Definition at line 24 of file stdair_exceptions.hpp.

32.132.2.2 stdair::RootException::RootException () [inline]

Default constructor.

Definition at line 28 of file stdair_exceptions.hpp.

32.132.2.3 virtual stdair::RootException::~RootException () throw () [inline, virtual]

Destructor.

Definition at line 33 of file stdair_exceptions.hpp.

32.132.3 Member Function Documentation

32.132.3.1 const char* stdair::RootException::what () const throw () [inline]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References _what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.132.4 Member Data Documentation

32.132.4.1 std::string stdair::RootException::_what [protected]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by what().

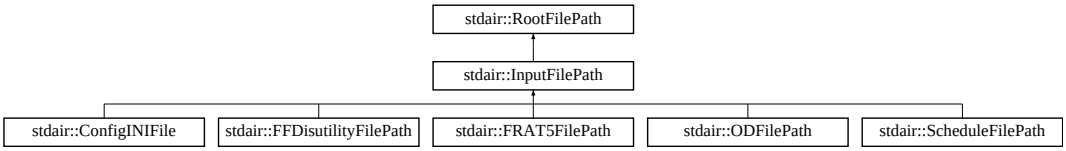
The documentation for this class was generated from the following file:

- stdair/stdair_exceptions.hpp

32.133 stdair::RootFilePath Class Reference

Root of the input and output files.

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::RootFilePath::



Public Member Functions

- RootFilePath (const Filename_T &iFilename)
- RootFilePath ()
- virtual ~RootFilePath ()
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.133.1 Detailed Description

Root of the input and output files. All the files inherit from that root.

Definition at line 22 of file stdair_file.hpp.

32.133.2 Constructor & Destructor Documentation

32.133.2.1 stdair::RootFilePath::RootFilePath (const Filename_T & *iFilename*) `[inline]`

Main Constructor.

Definition at line 27 of file stdair_file.hpp.

32.133.2.2 stdair::RootFilePath::RootFilePath () `[inline]`

Default constructor.

Definition at line 32 of file stdair_file.hpp.

32.133.2.3 virtual stdair::RootFilePath::~RootFilePath () `[inline, virtual]`

Destructor.

Definition at line 37 of file stdair_file.hpp.

32.133.3 Member Function Documentation

32.133.3.1 const char* stdair::RootFilePath::name () const [inline]

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References _filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.133.4 Member Data Documentation

32.133.4.1 const Filename_T stdair::RootFilePath::_filename [protected]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by name().

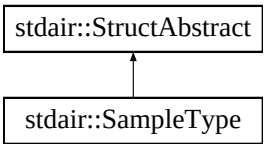
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

32.134 stdair::SampleType Struct Reference

Enumeration of BOM sample types.

`#include <stdair/basic/SampleType.hpp>`Inheritance diagram for stdair::SampleType::



Public Types

- enum EN_SampleType {
  ALL = 0, A4P, RMS, INV,
  SCH, RAC, FQT, CRS,
  DEM, EVT, CCM, LAST_VALUE }

Public Member Functions

- EN_SampleType getType () const
- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_SampleType &) const
- SampleType (const EN_SampleType &)

- SampleType (const char iType)
- SampleType (const SampleType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_SampleType &)
- static char getTypeLabel (const EN_SampleType &)
- static std::string getTypeLabelAsString (const EN_SampleType &)
- static std::string describeLabels ()

32.134.1 Detailed Description

Enumeration of BOM sample types. In order to test some components, it is often easier to fill the BOM tree with hard-coded structures than set up CSV input files and parsing them. That enumeration structure tells for which component(s) the sample BOM tree should be built. By default, a BOM sample tree is built for all the components, i.e., it contains StdAir objects for all the other components (AirInv, AirSched, etc).

Definition at line 25 of file SampleType.hpp.

32.134.2 Member Enumeration Documentation

32.134.2.1 enum stdair::SampleType::EN_SampleType

Enumerator:

ALL
A4P
RMS
INV
SCH
RAC
FQT
CRS
DEM
EVT
CCM
LAST_VALUE

Definition at line 27 of file SampleType.hpp.

32.134.3 Constructor & Destructor Documentation

32.134.3.1 stdair::SampleType::SampleType (const EN_SampleType & *iSampleType*)

Constructor.

Definition at line 36 of file SampleType.cpp.

32.134.3.2 stdair::SampleType::SampleType (const char *iType*)

Constructor.

Definition at line 41 of file SampleType.cpp.

References A4P, ALL, CCM, CRS, DEM, describeLabels(), EVT, FQT, INV, LAST_VALUE, RAC, RMS, and SCH.

32.134.3.3 stdair::SampleType::SampleType (const SampleType & *iSampleType*)

Default copy constructor.

Definition at line 31 of file SampleType.cpp.

32.134.4 Member Function Documentation

32.134.4.1 const std::string & stdair::SampleType::getLabel (const EN_SampleType & *iType*) `[static]`

Get the label as a string (e.g., "Inventory" or "Schedule").

Definition at line 67 of file SampleType.cpp.

32.134.4.2 char stdair::SampleType::getTypeLabel (const EN_SampleType & *iType*) `[static]`

Get the label as a single char (e.g., 'I' or 'S').

Definition at line 72 of file SampleType.cpp.

32.134.4.3 std::string stdair::SampleType::getTypeLabelAsString (const EN_SampleType & *iType*) `[static]`

Get the label as a string of a single char (e.g., "I" or "S").

Definition at line 77 of file SampleType.cpp.

32.134.4.4 std::string stdair::SampleType::describeLabels () `[static]`

List the labels.

Definition at line 84 of file SampleType.cpp.

References LAST_VALUE.

Referenced by SampleType().

32.134.4.5 SampleType::EN_SampleType stdair::SampleType::getType () const

Get the enumerated value.

Definition at line 96 of file SampleType.cpp.

32.134.4.6 std::string stdair::SampleType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "I" or "S").

Definition at line 101 of file SampleType.cpp.

32.134.4.7 const std::string stdair::SampleType::describe () const [virtual]

Give a description of the structure (e.g., "Inventory" or "Schedule").

Implements stdair::StructAbstract.

Definition at line 108 of file SampleType.cpp.

32.134.4.8 bool stdair::SampleType::operator== (const EN_SampleType & *iType*) const

Comparison operator.

Definition at line 115 of file SampleType.cpp.

32.134.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.134.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual, inherited]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

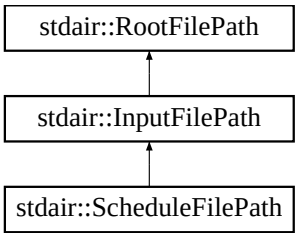
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/SampleType.hpp
- stdair/basic/SampleType.cpp

32.135 stdair::ScheduleFilePath Class Reference

`#include <stdair/stdair_file.hpp>`Inheritance diagram for stdair::ScheduleFilePath::



Public Member Functions

- ScheduleFilePath (const Filename_T &iFilename)
- const char * name () const

Protected Attributes

- const Filename_T _filename

32.135.1 Detailed Description

Schedule input file.

Definition at line 64 of file stdair_file.hpp.

32.135.2 Constructor & Destructor Documentation

32.135.2.1 stdair::ScheduleFilePath::ScheduleFilePath (const Filename_T & *iFilename*) `[inline, explicit]`

Constructor.

Definition at line 69 of file stdair_file.hpp.

32.135.3 Member Function Documentation

32.135.3.1 const char* stdair::RootFilePath::name () const `[inline, inherited]`

Give the details of the exception.

Definition at line 42 of file stdair_file.hpp.

References stdair::RootFilePath::_filename.

Referenced by stdair::BomINIImport::importINIConfig().

32.135.4 Member Data Documentation

32.135.4.1 const Filename_T stdair::RootFilePath::_filename [protected, inherited]

Name of the file.

Definition at line 50 of file stdair_file.hpp.

Referenced by stdair::RootFilePath::name().

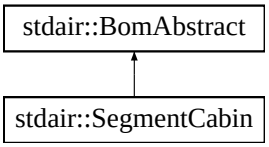
The documentation for this class was generated from the following file:

- stdair/stdair_file.hpp

32.136 stdair::SegmentCabin Class Reference

Class representing the actual attributes for an airline segment-cabin.

`#include <stdair/bom/SegmentCabin.hpp>`Inheritance diagram for stdair::SegmentCabin::



Public Types

- typedef SegmentCabinKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const CabinCode_T & getCabinCode () const
- const MapKey_T getFullerKey () const
- const SegmentSnapshotTable & getSegmentSnapshotTable () const
- const CabinCapacity_T & getCapacity () const
- const BlockSpace_T & getBlockSpace () const
- const BlockSpace_T & getMIN () const
- const UPR_T & getUPR () const
- const NbOfBookings_T & getBookingCounter () const
- const CommittedSpace_T & getCommittedSpace () const
- const Availability_T & getAvailabilityPool () const
- const BidPrice_T & getCurrentBidPrice () const

- const BidPriceVector_T & getBidPriceVector () const
- const bool getFareFamilyStatus () const
- const PolicyList_T & getConvexHull () const
- void setSegmentSnapshotTable (SegmentSnapshotTable &ioTable)
- void setCapacity (const CabinCapacity_T &iCapacity)
- void setBlockSpace (const BlockSpace_T &iBlockSpace)
- void setMIN (const BlockSpace_T &iMIN)
- void setUPR (const UPR_T &iUPR)
- void setBookingCounter (const NbOfBookings_T &iBookingCounter)
- void setCommittedSpace (const CommittedSpace_T &iCommittedSpace)
- void setAvailabilityPool (const Availability_T &iAvailabilityPool)
- void setBidPriceVector (const BidPriceVector_T &iBPV)
- void activateFareFamily ()
- void updateFromReservation (const NbOfBookings_T &)
- void resetConvexHull ()
- void addPolicy (Policy &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const std::string describeConvexHull () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentCabin (const Key_T &)
- virtual ~SegmentCabin ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentSnapshotTable * _segmentSnapshotTable
- CabinCapacity_T _capacity
- BlockSpace_T _blockSpace
- BlockSpace_T _min
- UPR_T _upr
- NbOfBookings_T _bookingCounter
- CommittedSpace_T _committedSpace
- Availability_T _availabilityPool
- BidPriceVector_T _bidPriceVector
- BidPrice_T _currentBidPrice
- bool _fareFamilyActivation
- PolicyList_T _convexHull

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.136.1 Detailed Description

Class representing the actual attributes for an airline segment-cabin.

Definition at line 33 of file SegmentCabin.hpp.

32.136.2 Member Typedef Documentation

32.136.2.1 typedef SegmentCabinKey stdair::SegmentCabin::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 44 of file SegmentCabin.hpp.

32.136.3 Constructor & Destructor Documentation

32.136.3.1 stdair::SegmentCabin::SegmentCabin (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 39 of file SegmentCabin.cpp.

32.136.3.2 stdair::SegmentCabin::∼SegmentCabin () `[protected, virtual]`

Destructor.

Definition at line 52 of file SegmentCabin.cpp.

32.136.4 Member Function Documentation

32.136.4.1 const Key_T& stdair::SegmentCabin::getKey () const `[inline]`

Get the segment-cabin key (cabin code).

Definition at line 52 of file SegmentCabin.hpp.

References _key.

32.136.4.2 BomAbstract∗ const stdair::SegmentCabin::getParent () const `[inline]`

Get the parent object.

Definition at line 59 of file SegmentCabin.hpp.

References _parent.

32.136.4.3 const HolderMap_T& stdair::SegmentCabin::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 66 of file SegmentCabin.hpp.

References _holderMap.

32.136.4.4 const CabinCode_T& stdair::SegmentCabin::getCabinCode () const [inline]

Get the cabin code (primary key).

Definition at line 73 of file SegmentCabin.hpp.

References _key, and stdair::SegmentCabinKey::getCabinCode().

Referenced by getFullerKey().

32.136.4.5 const MapKey_T stdair::SegmentCabin::getFullerKey () const

Get the (segment-date, segment-cabin) key (board point, off point and cabin code).

Note:

That method assumes that the parent object derives from the SegmentDate class, as it needs to have access to the describeKey() method.

Definition at line 56 of file SegmentCabin.cpp.

References stdair::DEFAULT_KEY_FLD_DELIMITER, stdair::SegmentDate::describeKey(), and getCabinCode().

32.136.4.6 const SegmentSnapshotTable& stdair::SegmentCabin::getSegmentSnapshotTable () const [inline]

Get the guilltine block.

Definition at line 88 of file SegmentCabin.hpp.

References _segmentSnapshotTable.

32.136.4.7 const CabinCapacity_T& stdair::SegmentCabin::getCapacity () const [inline]

Get the cabin capacity.

Definition at line 94 of file SegmentCabin.hpp.

References _capacity.

32.136.4.8 const BlockSpace_T& stdair::SegmentCabin::getBlockSpace () const [inline]

Get the blocked number of bookings.

Definition at line 99 of file SegmentCabin.hpp.

References _blockSpace.

32.136.4.9 const BlockSpace_T& stdair::SegmentCabin::getMIN () const [inline]

Get the blocked number of bookings.

Definition at line 104 of file SegmentCabin.hpp.

References _min.

32.136.4.10 const UPR_T& stdair::SegmentCabin::getUPR () const [inline]

Unsold Protection (UPR).

Definition at line 109 of file SegmentCabin.hpp.

References _upr.

32.136.4.11 const NbOfBookings_T& stdair::SegmentCabin::getBookingCounter () const [inline]

Get the booking counter.

Definition at line 114 of file SegmentCabin.hpp.

References _bookingCounter.

32.136.4.12 const CommittedSpace_T& stdair::SegmentCabin::getCommittedSpace () const [inline]

Get the committed Space value.

Definition at line 119 of file SegmentCabin.hpp.

References _committedSpace.

32.136.4.13 const Availability_T& stdair::SegmentCabin::getAvailabilityPool () const [inline]

Get the availability pool value.

Definition at line 124 of file SegmentCabin.hpp.

References _availabilityPool.

32.136.4.14 const BidPrice_T& stdair::SegmentCabin::getCurrentBidPrice () const [inline]

Retrieve the current Bid-Price.

Definition at line 129 of file SegmentCabin.hpp.

References _currentBidPrice.

32.136.4.15 const BidPriceVector_T& stdair::SegmentCabin::getBidPriceVector () const [inline]

Retrieve the Bid-Price Vector.

Definition at line 134 of file SegmentCabin.hpp.

References _bidPriceVector.

32.136.4.16 const bool stdair::SegmentCabin::getFareFamilyStatus () const [inline]

Retrieve the status of fare family.

Definition at line 139 of file SegmentCabin.hpp.

References _fareFamilyActivation.

32.136.4.17 const PolicyList_T& stdair::SegmentCabin::getConvexHull () const [inline]

Retrieve the convex hull.

Definition at line 144 of file SegmentCabin.hpp.

References _convexHull.

32.136.4.18 void stdair::SegmentCabin::setSegmentSnapshotTable (SegmentSnapshotTable & *ioTable*) [inline]

Set the snapshot table.

Definition at line 151 of file SegmentCabin.hpp.

References _segmentSnapshotTable.

32.136.4.19 void stdair::SegmentCabin::setCapacity (const CabinCapacity_T & *iCapacity*) [inline]

Set the cabin capacity.

Definition at line 156 of file SegmentCabin.hpp.

References _capacity.

32.136.4.20 void stdair::SegmentCabin::setBlockSpace (const BlockSpace_T & *iBlockSpace*) [inline]

Set the blocked number of seats.

Definition at line 161 of file SegmentCabin.hpp.

References _blockSpace.

32.136.4.21 void stdair::SegmentCabin::setMIN (const BlockSpace_T & *iMIN*) [inline]

Set the blocked number of seats.

Definition at line 166 of file SegmentCabin.hpp.

References _min.

32.136.4.22 void stdair::SegmentCabin::setUPR (const UPR_T & *iUPR*) [inline]

Set the Unsold Protection (UPR).

Definition at line 171 of file SegmentCabin.hpp.

References _upr.

32.136.4.23 void stdair::SegmentCabin::setBookingCounter (const NbOfBookings_T & *iBookingCounter*) [inline]

Set the total number of bookings.

Definition at line 176 of file SegmentCabin.hpp.

References _bookingCounter.

32.136.4.24 void stdair::SegmentCabin::setCommittedSpace (const CommittedSpace_T & *iCommittedSpace*) [inline]

Set the value of committed space.

Definition at line 181 of file SegmentCabin.hpp.

References _committedSpace.

32.136.4.25 void stdair::SegmentCabin::setAvailabilityPool (const Availability_T & *iAvailabilityPool*) [inline]

Set the value of availability pool.

Definition at line 186 of file SegmentCabin.hpp.

References _availabilityPool.

32.136.4.26 void stdair::SegmentCabin::setBidPriceVector (const BidPriceVector_T & *iBPV*) [inline]

Set the Bid-Price Vector.

Definition at line 191 of file SegmentCabin.hpp.

References _bidPriceVector.

32.136.4.27 void stdair::SegmentCabin::activateFareFamily () [inline]

Activate fare family.

Definition at line 196 of file SegmentCabin.hpp.

References _fareFamilyActivation.

32.136.4.28 void stdair::SegmentCabin::updateFromReservation (const NbOfBookings_T & *iNbOfBookings*)

Register a sale.

Definition at line 85 of file SegmentCabin.cpp.

References _committedSpace.

32.136.4.29 void stdair::SegmentCabin::resetConvexHull () [inline]

Reset the convex hull.

Definition at line 206 of file SegmentCabin.hpp.

References _convexHull.

32.136.4.30 void stdair::SegmentCabin::addPolicy (Policy & *ioPolicy*)

Add a policy to the convex hull. Note: we do not use the FacBomManager here because the convex hull is not a list of children but a temporary list of policies.

Definition at line 90 of file SegmentCabin.cpp.

References _convexHull.

32.136.4.31 void stdair::SegmentCabin::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 221 of file SegmentCabin.hpp.

References toString().

32.136.4.32 void stdair::SegmentCabin::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 230 of file SegmentCabin.hpp.

32.136.4.33 std::string stdair::SegmentCabin::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 65 of file SegmentCabin.cpp.

References describeKey().

Referenced by toStream().

32.136.4.34 const std::string stdair::SegmentCabin::describeKey () const [inline]

Get a string describing the key.

Definition at line 241 of file SegmentCabin.hpp.

References _key, and stdair::SegmentCabinKey::toString().

Referenced by toString().

32.136.4.35 const std::string stdair::SegmentCabin::describeConvexHull () const

Get a string describing the convex hull.

Definition at line 72 of file SegmentCabin.cpp.

References _convexHull, and stdair::Policy::toString().

32.136.4.36 template<class Archive > void stdair::SegmentCabin::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 229 of file CmdBomSerialiser.cpp.

References _key.

32.136.5 Friends And Related Function Documentation

32.136.5.1 friend class FacBom [friend]

Definition at line 34 of file SegmentCabin.hpp.

32.136.5.2 friend class FacCloneBom [friend]

Definition at line 35 of file SegmentCabin.hpp.

32.136.5.3 friend class FacBomManager [friend]

Definition at line 36 of file SegmentCabin.hpp.

32.136.5.4 friend class boost::serialization::access [friend]

Definition at line 37 of file SegmentCabin.hpp.

32.136.6 Member Data Documentation

32.136.6.1 Key_T stdair::SegmentCabin::_key [protected]

Primary key (cabin code).

Definition at line 300 of file SegmentCabin.hpp.

Referenced by describeKey(), getCabinCode(), getKey(), and serialize().

32.136.6.2 BomAbstract* stdair::SegmentCabin::_parent [protected]

Pointer on the parent class (SegmentDate).

Definition at line 305 of file SegmentCabin.hpp.

Referenced by getParent().

32.136.6.3 HolderMap_T stdair::SegmentCabin::_holderMap [protected]

Map holding the children (FareFamily or BookingClass objects).

Definition at line 310 of file SegmentCabin.hpp.

Referenced by getHolderMap().

32.136.6.4 SegmentSnapshotTable* stdair::SegmentCabin::_segmentSnapshotTable [protected]

The data table used for Revenue Management activities.

Definition at line 315 of file SegmentCabin.hpp.

Referenced by getSegmentSnapshotTable(), and setSegmentSnapshotTable().

32.136.6.5 CabinCapacity_T stdair::SegmentCabin::_capacity [protected]

Capacity of the cabin.

Definition at line 318 of file SegmentCabin.hpp.

Referenced by getCapacity(), and setCapacity().

32.136.6.6 BlockSpace_T stdair::SegmentCabin::_blockSpace [protected]

Blocked capacity.

Definition at line 321 of file SegmentCabin.hpp.

Referenced by getBlockSpace(), and setBlockSpace().

32.136.6.7 BlockSpace_T stdair::SegmentCabin::_min [protected]

Blocked number of seats.

Definition at line 324 of file SegmentCabin.hpp.

Referenced by getMIN(), and setMIN().

32.136.6.8 UPR_T stdair::SegmentCabin::_upr [protected]

Unsold Protection (UPR).

Definition at line 327 of file SegmentCabin.hpp.

Referenced by getUPR(), and setUPR().

32.136.6.9 NbOfBookings_T stdair::SegmentCabin::_bookingCounter [protected]

Aggregated number of bookings.

Definition at line 330 of file SegmentCabin.hpp.

Referenced by getBookingCounter(), and setBookingCounter().

32.136.6.10 CommittedSpace_T stdair::SegmentCabin::_committedSpace [protected]

Aggreated committed space.

Definition at line 333 of file SegmentCabin.hpp.

Referenced by getCommittedSpace(), setCommittedSpace(), and updateFromReservation().

32.136.6.11 Availability_T stdair::SegmentCabin::_availabilityPool [protected]

Aggregated availability pool.

Definition at line 336 of file SegmentCabin.hpp.

Referenced by getAvailabilityPool(), and setAvailabilityPool().

32.136.6.12 BidPriceVector_T stdair::SegmentCabin::_bidPriceVector [protected]

Bid-Price Vector (BPV).

Definition at line 339 of file SegmentCabin.hpp.

Referenced by getBidPriceVector(), and setBidPriceVector().

32.136.6.13 BidPrice_T stdair::SegmentCabin::_currentBidPrice [protected]

Current Bid-Price (BP).

Definition at line 342 of file SegmentCabin.hpp.

Referenced by getCurrentBidPrice().

32.136.6.14 bool stdair::SegmentCabin::_fareFamilyActivation [protected]

Indicate if fare family is in use.

Definition at line 345 of file SegmentCabin.hpp.

Referenced by activateFareFamily(), and getFareFamilyStatus().

32.136.6.15 PolicyList_T stdair::SegmentCabin::_convexHull [protected]

The convex hull of MRT.

Definition at line 348 of file SegmentCabin.hpp.

Referenced by addPolicy(), describeConvexHull(), getConvexHull(), and resetConvexHull().

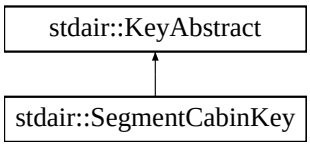
The documentation for this class was generated from the following files:

- stdair/bom/SegmentCabin.hpp
- stdair/bom/SegmentCabin.cpp
- stdair/command/CmdBomSerialiser.cpp

32.137 stdair::SegmentCabinKey Struct Reference

Key of a given segment-cabin, made of a cabin code (only).

`#include <stdair/bom/SegmentCabinKey.hpp>`Inheritance diagram for stdair::SegmentCabinKey::



Public Member Functions

- SegmentCabinKey (const CabinCode_T &iCabinCode)
- SegmentCabinKey (const SegmentCabinKey &)
- ~SegmentCabinKey ()
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.137.1 Detailed Description

Key of a given segment-cabin, made of a cabin code (only).

Definition at line 26 of file SegmentCabinKey.hpp.

32.137.2 Constructor & Destructor Documentation

32.137.2.1 stdair::SegmentCabinKey::SegmentCabinKey (const CabinCode_T & *iCabinCode*)

Constructor.

Definition at line 23 of file SegmentCabinKey.cpp.

32.137.2.2 stdair::SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey & *iKey*)

Copy constructor.

Definition at line 28 of file SegmentCabinKey.cpp.

32.137.2.3 stdair::SegmentCabinKey::~SegmentCabinKey ()

Destructor.

Definition at line 33 of file SegmentCabinKey.cpp.

32.137.3 Member Function Documentation

32.137.3.1 const CabinCode_T& stdair::SegmentCabinKey::getCabinCode () const `[inline]`

Get the cabin code.

Definition at line 56 of file SegmentCabinKey.hpp.

Referenced by stdair::SegmentCabin::getCabinCode().

32.137.3.2 void stdair::SegmentCabinKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 37 of file SegmentCabinKey.cpp.

References toString().

32.137.3.3 void stdair::SegmentCabinKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 42 of file SegmentCabinKey.cpp.

32.137.3.4 const std::string stdair::SegmentCabinKey::toString () const `[virtual]`

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 46 of file SegmentCabinKey.cpp.

Referenced by stdair::SegmentCabin::describeKey(), stdair::BomRetriever::retrieveDummySegmentCabin(), and toStream().

32.137.3.5 template<class Archive > void stdair::SegmentCabinKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 68 of file SegmentCabinKey.cpp.

32.137.4 Friends And Related Function Documentation

32.137.4.1 friend class boost::serialization::access `[friend]`

Definition at line 27 of file SegmentCabinKey.hpp.

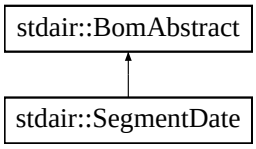
The documentation for this struct was generated from the following files:

- stdair/bom/SegmentCabinKey.hpp
- stdair/bom/SegmentCabinKey.cpp

32.138 stdair::SegmentDate Class Reference

Class representing the actual attributes for an airline segment-date.

`#include <stdair/bom/SegmentDate.hpp>`Inheritance diagram for stdair::SegmentDate::



Public Types

- typedef SegmentDateKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- const HolderMap_T & getHolderMap () const
- const Date_T & getBoardingDate () const
- const Duration_T & getBoardingTime () const
- const Date_T & getOffDate () const
- const Duration_T & getOffTime () const
- const Duration_T & getElapsedTime () const
- const Distance_T & getDistance () const
- const DateOffset_T getDateOffset () const
- const Duration_T getTimeOffset () const
- SegmentDate * getOperatingSegmentDate () const
- const SegmentDateList_T & getMarketingSegmentDateList () const
- const RoutingLegKeyList_T & getLegKeyList () const
- void setBoardingDate (const Date_T &iBoardingDate)
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffDate (const Date_T &iOffDate)
- void setOffTime (const Duration_T &iOffTime)
- void setElapsedTime (const Duration_T &iElapsedTime)
- void setDistance (const Distance_T &iDistance)
- void addLegKey (const std::string &iLegKey)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentDate (const Key_T &)
- virtual ~SegmentDate ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentDate * _operatingSegmentDate
- SegmentDateList_T _marketingSegmentDateList
- Date_T _boardingDate
- Duration_T _boardingTime

- Date_T _offDate
- Duration_T _offTime
- Duration_T _elapsedTime
- Distance_T _distance
- RoutingLegKeyList_T _routingLegKeyList

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager
- class boost::serialization::access

32.138.1 Detailed Description

Class representing the actual attributes for an airline segment-date.

Definition at line 36 of file SegmentDate.hpp.

32.138.2 Member Typedef Documentation

32.138.2.1 typedef SegmentDateKey stdair::SegmentDate::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 47 of file SegmentDate.hpp.

32.138.3 Constructor & Destructor Documentation

32.138.3.1 stdair::SegmentDate::SegmentDate (const Key_T & *iKey*) `[protected]`

Constructor.

Definition at line 38 of file SegmentDate.cpp.

32.138.3.2 stdair::SegmentDate::~SegmentDate () `[protected, virtual]`

Destructor.

Definition at line 44 of file SegmentDate.cpp.

32.138.4 Member Function Documentation

32.138.4.1 const Key_T& stdair::SegmentDate::getKey () const `[inline]`

Get the segment-date key.

Definition at line 55 of file SegmentDate.hpp.

References _key.

32.138.4.2 BomAbstract* const stdair::SegmentDate::getParent () const [inline]

Get the parent object.

Definition at line 62 of file SegmentDate.hpp.

References _parent.

32.138.4.3 const AirportCode_T& stdair::SegmentDate::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 69 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::getBoardingPoint().

32.138.4.4 const AirportCode_T& stdair::SegmentDate::getOffPoint () const [inline]

Get the off point (part of the primary key).

Definition at line 76 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::getOffPoint().

32.138.4.5 const HolderMap_T& stdair::SegmentDate::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 83 of file SegmentDate.hpp.

References _holderMap.

32.138.4.6 const Date_T& stdair::SegmentDate::getBoardingDate () const [inline]

Get the boarding date.

Definition at line 90 of file SegmentDate.hpp.

References _boardingDate.

32.138.4.7 const Duration_T& stdair::SegmentDate::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 97 of file SegmentDate.hpp.

References _boardingTime.

32.138.4.8 const Date_T& stdair::SegmentDate::getOffDate () const [inline]

Get the off date.

Definition at line 104 of file SegmentDate.hpp.

References _offDate.

32.138.4.9 const Duration_T& stdair::SegmentDate::getOffTime () const [inline]

Get the off time.

Definition at line 111 of file SegmentDate.hpp.

References _offTime.

32.138.4.10 const Duration_T& stdair::SegmentDate::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 118 of file SegmentDate.hpp.

References _elapsedTime.

32.138.4.11 const Distance_T& stdair::SegmentDate::getDistance () const [inline]

Get the distance.

Definition at line 125 of file SegmentDate.hpp.

References _distance.

32.138.4.12 const DateOffset_T stdair::SegmentDate::getDateOffset () const [inline]

Get the date offset (off date - boarding date).

Definition at line 132 of file SegmentDate.hpp.

References _boardingDate, and _offDate.

Referenced by getTimeOffset().

32.138.4.13 const Duration_T stdair::SegmentDate::getTimeOffset () const

Get the time offset between boarding and off points.

It is defined as being:

TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) $*$ 24

- ElapsedTime.

Definition at line 55 of file SegmentDate.cpp.

References _boardingTime, _elapsedTime, _offTime, and getDateOffset().

32.138.4.14 SegmentDate* stdair::SegmentDate::getOperatingSegmentDate () const [inline]

Get the "operating" segment date.

Definition at line 149 of file SegmentDate.hpp.

References _operatingSegmentDate.

32.138.4.15 const SegmentDateList_T& stdair::SegmentDate::getMarketingSegmentDateList () const [inline]

Get the list of marketing segment dates.

Definition at line 156 of file SegmentDate.hpp.

References _marketingSegmentDateList.

32.138.4.16 const RoutingLegKeyList_T& stdair::SegmentDate::getLegKeyList () const [inline]

Get the list of routing leg keys.

Definition at line 163 of file SegmentDate.hpp.

References _routingLegKeyList.

32.138.4.17 void stdair::SegmentDate::setBoardingDate (const Date_T & *iBoardingDate*) [inline]

Set the boarding date.

Definition at line 172 of file SegmentDate.hpp.

References _boardingDate.

32.138.4.18 void stdair::SegmentDate::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 179 of file SegmentDate.hpp.

References _boardingTime.

32.138.4.19 void stdair::SegmentDate::setOffDate (const Date_T & *iOffDate*) [inline]

Set the off date.

Definition at line 186 of file SegmentDate.hpp.

References _offDate.

32.138.4.20 void stdair::SegmentDate::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 193 of file SegmentDate.hpp.

References _offTime.

32.138.4.21 void stdair::SegmentDate::setElapsedTime (const Duration_T & *iElapsedTime*) [inline]

Set the elapsed time.

Definition at line 200 of file SegmentDate.hpp.

References _elapsedTime.

32.138.4.22 void stdair::SegmentDate::setDistance (const Distance_T & *iDistance*) [inline]

Set the distance.

Definition at line 207 of file SegmentDate.hpp.

References _distance.

32.138.4.23 void stdair::SegmentDate::addLegKey (const std::string & *iLegKey*) [inline]

Add a routing leg key to the list.

Definition at line 214 of file SegmentDate.hpp.

References _routingLegKeyList.

32.138.4.24 void stdair::SegmentDate::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 233 of file SegmentDate.hpp.

References toString().

32.138.4.25 void stdair::SegmentDate::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 242 of file SegmentDate.hpp.

32.138.4.26 std::string stdair::SegmentDate::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 48 of file SegmentDate.cpp.

References describeKey().

Referenced by toStream().

32.138.4.27 const std::string stdair::SegmentDate::describeKey () const [inline]

Get a string describing the key.

Definition at line 253 of file SegmentDate.hpp.

References _key, and stdair::SegmentDateKey::toString().

Referenced by stdair::SegmentCabin::getFullerKey(), stdair::BomRetriever::retrieveFullKeyFromSegmentDate(), and toString().

32.138.4.28 template<class Archive > void stdair::SegmentDate::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 208 of file CmdBomSerialiser.cpp.

References _key.

32.138.5 Friends And Related Function Documentation

32.138.5.1 friend class FacBom [friend]

Definition at line 37 of file SegmentDate.hpp.

32.138.5.2 friend class FacCloneBom [friend]

Definition at line 38 of file SegmentDate.hpp.

32.138.5.3 friend class FacBomManager [friend]

Definition at line 39 of file SegmentDate.hpp.

32.138.5.4 friend class boost::serialization::access [friend]

Definition at line 40 of file SegmentDate.hpp.

32.138.6 Member Data Documentation

32.138.6.1 Key_T stdair::SegmentDate::_key [protected]

Primary key (origin and destination).

Definition at line 307 of file SegmentDate.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), getOffPoint(), and serialize().

32.138.6.2 BomAbstract* stdair::SegmentDate::_parent [protected]

Pointer on the parent class (FlightDate).

Definition at line 312 of file SegmentDate.hpp.

Referenced by getParent().

32.138.6.3 HolderMap_T stdair::SegmentDate::_holderMap [protected]

Map holding the children (SegmentCabin objects).

Definition at line 317 of file SegmentDate.hpp.

Referenced by getHolderMap().

32.138.6.4 SegmentDate* stdair::SegmentDate::_operatingSegmentDate [protected]

Pointer on the operating SegmentDate. Nota: 1. "operating" refers to the codeshare contract seller. 2. the pointer will be NULL if the segment date is itself the "operating" one.

Definition at line 325 of file SegmentDate.hpp.

Referenced by getOperatingSegmentDate().

32.138.6.5 SegmentDateList_T stdair::SegmentDate::_marketingSegmentDateList [protected]

List holding the marketing segment dates. Nota: 1. "marketing" refers to the codeshare contract seller. 2. the list will be empty if the segment date is itself the "marketing" one.

Definition at line 333 of file SegmentDate.hpp.

Referenced by getMarketingSegmentDateList().

32.138.6.6 Date_T stdair::SegmentDate::_boardingDate [protected]

Boarding date.

Definition at line 338 of file SegmentDate.hpp.

Referenced by getBoardingDate(), getDateOffset(), and setBoardingDate().

32.138.6.7 Duration_T stdair::SegmentDate::_boardingTime [protected]

Boarding time.

Definition at line 343 of file SegmentDate.hpp.

Referenced by getBoardingTime(), getTimeOffset(), and setBoardingTime().

32.138.6.8 Date_T stdair::SegmentDate::_offDate [protected]

Landing date.

Definition at line 348 of file SegmentDate.hpp.

Referenced by getDateOffset(), getOffDate(), and setOffDate().

32.138.6.9 Duration_T stdair::SegmentDate::_offTime [protected]

Landing time.

Definition at line 353 of file SegmentDate.hpp.

Referenced by getOffTime(), getTimeOffset(), and setOffTime().

32.138.6.10 Duration_T stdair::SegmentDate::_elapsedTime [protected]

Trip elapsed time.

Definition at line 358 of file SegmentDate.hpp.

Referenced by getElapsedTime(), getTimeOffset(), and setElapsedTime().

32.138.6.11 Distance_T stdair::SegmentDate::_distance [protected]

Trip distance.

Definition at line 363 of file SegmentDate.hpp.

Referenced by getDistance(), and setDistance().

32.138.6.12 RoutingLegKeyList_T stdair::SegmentDate::_routingLegKeyList [protected]

List of routing leg keys.

Definition at line 368 of file SegmentDate.hpp.

Referenced by addLegKey(), and getLegKeyList().

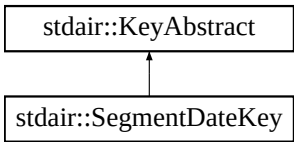
The documentation for this class was generated from the following files:

- stdair/bom/SegmentDate.hpp
- stdair/bom/SegmentDate.cpp
- stdair/command/CmdBomSerialiser.cpp

32.139 stdair::SegmentDateKey Struct Reference

Key of a given segment-date, made of an origin and a destination airports.

`#include <stdair/bom/SegmentDateKey.hpp>`Inheritance diagram for stdair::SegmentDateKey::



Public Member Functions

- SegmentDateKey (const AirportCode_T &, const AirportCode_T &)
- SegmentDateKey (const SegmentDateKey &)
- ~SegmentDateKey ()
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.139.1 Detailed Description

Key of a given segment-date, made of an origin and a destination airports.

Definition at line 24 of file SegmentDateKey.hpp.

32.139.2 Constructor & Destructor Documentation

32.139.2.1 stdair::SegmentDateKey::SegmentDateKey (const AirportCode_T & *iBoardingPoint*, const AirportCode_T & *iOffPoint*)

Main constructor.

Definition at line 25 of file SegmentDateKey.cpp.

32.139.2.2 stdair::SegmentDateKey::SegmentDateKey (const SegmentDateKey & *iKey*)

Copy constructor.

Definition at line 31 of file SegmentDateKey.cpp.

32.139.2.3 stdair::SegmentDateKey::~SegmentDateKey ()

Destructor.

Definition at line 36 of file SegmentDateKey.cpp.

32.139.3 Member Function Documentation

32.139.3.1 const AirportCode_T& stdair::SegmentDateKey::getBoardingPoint () const [inline]

Get the boardinging point.

Definition at line 51 of file SegmentDateKey.hpp.

Referenced by stdair::SegmentDate::getBoardingPoint(), and stdair::OnDDateKey::getOrigin().

32.139.3.2 const AirportCode_T& stdair::SegmentDateKey::getOffPoint () const [inline]

Get the arrival point.

Definition at line 56 of file SegmentDateKey.hpp.

Referenced by stdair::OnDDateKey::getDestination(), and stdair::SegmentDate::getOffPoint().

32.139.3.3 void stdair::SegmentDateKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file SegmentDateKey.cpp.

References toString().

32.139.3.4 void stdair::SegmentDateKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file SegmentDateKey.cpp.

32.139.3.5 const std::string stdair::SegmentDateKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file SegmentDateKey.cpp.

References stdair::DEFAULT_KEY_SUB_FLD_DELIMITER.

Referenced by stdair::SegmentDate::describeKey(), stdair::FlightDate::getSegmentDate(), stdair::BomRetriever::retrieveSegmentDateFromLongKey(), and toStream().

32.139.3.6 template<class Archive > void stdair::SegmentDateKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) `[inline]`

Serialisation.

Definition at line 72 of file SegmentDateKey.cpp.

32.139.4 Friends And Related Function Documentation

32.139.4.1 friend class boost::serialization::access `[friend]`

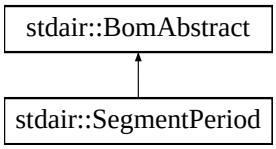
Definition at line 25 of file SegmentDateKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/SegmentDateKey.hpp
- stdair/bom/SegmentDateKey.cpp

32.140 stdair::SegmentPeriod Class Reference

`#include <stdair/bom/SegmentPeriod.hpp>`Inheritance diagram for stdair::SegmentPeriod::



Public Types

- typedef SegmentPeriodKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- const Duration_T & getBoardingTime () const
- const Duration_T & getOffTime () const
- const DateOffset_T & getBoardingDateOffset () const
- const DateOffset_T & getOffDateOffset () const
- const Duration_T & getElapsedTime () const

- const CabinBookingClassMap_T & getCabinBookingClassMap () const
- const HolderMap_T & getHolderMap () const
- void setBoardingTime (const Duration_T &iBoardingTime)
- void setOffTime (const Duration_T &iOffTime)
- void setBoardingDateOffset (const DateOffset_T &iDateOffset)
- void setOffDateOffset (const DateOffset_T &iDateOffset)
- void setElapsedTime (const Duration_T &iElapsedTime)
- void addCabinBookingClassList (const CabinCode_T &, const ClassList_String_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const

Protected Member Functions

- SegmentPeriod (const Key_T &)
- virtual ~SegmentPeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- Duration_T _boardingTime
- Duration_T _offTime
- DateOffset_T _boardingDateOffset
- DateOffset_T _offDateOffset
- Duration_T _elapsedTime
- CabinBookingClassMap_T _cabinBookingClassMap
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.140.1 Detailed Description

Class representing the actual attributes for an airline segment-period.

Definition at line 15 of file SegmentPeriod.hpp.

32.140.2 Member Typedef Documentation

32.140.2.1 typedef SegmentPeriodKey stdair::SegmentPeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 23 of file SegmentPeriod.hpp.

32.140.3 Constructor & Destructor Documentation

32.140.3.1 stdair::SegmentPeriod::SegmentPeriod (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 13 of file SegmentPeriod.cpp.

32.140.3.2 stdair::SegmentPeriod::~SegmentPeriod () [protected, virtual]

Destructor.

Definition at line 29 of file SegmentPeriod.cpp.

32.140.4 Member Function Documentation

32.140.4.1 const Key_T& stdair::SegmentPeriod::getKey () const [inline]

Get the segment-period key.

Definition at line 28 of file SegmentPeriod.hpp.

References _key.

32.140.4.2 BomAbstract* const stdair::SegmentPeriod::getParent () const [inline]

Get the parent object.

Definition at line 31 of file SegmentPeriod.hpp.

References _parent.

32.140.4.3 const AirportCode_T& stdair::SegmentPeriod::getBoardingPoint () const [inline]

Get the boarding point (part of the primary key).

Definition at line 34 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::getBoardingPoint().

32.140.4.4 const AirportCode_T& stdair::SegmentPeriod::getOffPoint () const [inline]

Get the off point (part of the primary key).

Definition at line 39 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::getOffPoint().

32.140.4.5 const Duration_T& stdair::SegmentPeriod::getBoardingTime () const [inline]

Get the boarding time.

Definition at line 42 of file SegmentPeriod.hpp.

References _boardingTime.

32.140.4.6 const Duration_T& stdair::SegmentPeriod::getOffTime () const [inline]

Get the off time.

Definition at line 45 of file SegmentPeriod.hpp.

References _offTime.

32.140.4.7 const DateOffset_T& stdair::SegmentPeriod::getBoardingDateOffset () const [inline]

Get the boarding date offset.

Definition at line 48 of file SegmentPeriod.hpp.

References _boardingDateOffset.

32.140.4.8 const DateOffset_T& stdair::SegmentPeriod::getOffDateOffset () const [inline]

Get the off date offset.

Definition at line 53 of file SegmentPeriod.hpp.

References _offDateOffset.

32.140.4.9 const Duration_T& stdair::SegmentPeriod::getElapsedTime () const [inline]

Get the elapsed time.

Definition at line 56 of file SegmentPeriod.hpp.

References _elapsedTime.

32.140.4.10 const CabinBookingClassMap_T& stdair::SegmentPeriod::getCabinBookingClassMap () const [inline]

Get the cabin booking class map.

Definition at line 59 of file SegmentPeriod.hpp.

References _cabinBookingClassMap.

32.140.4.11 const HolderMap_T& stdair::SegmentPeriod::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file SegmentPeriod.hpp.

References _holderMap.

32.140.4.12 void stdair::SegmentPeriod::setBoardingTime (const Duration_T & *iBoardingTime*) [inline]

Set the boarding time.

Definition at line 69 of file SegmentPeriod.hpp.

References _boardingTime.

32.140.4.13 void stdair::SegmentPeriod::setOffTime (const Duration_T & *iOffTime*) [inline]

Set the off time.

Definition at line 74 of file SegmentPeriod.hpp.

References _offTime.

32.140.4.14 void stdair::SegmentPeriod::setBoardingDateOffset (const DateOffset_T & *iDateOffset*) [inline]

Set the boarding date offset.

Definition at line 77 of file SegmentPeriod.hpp.

References _boardingDateOffset.

32.140.4.15 void stdair::SegmentPeriod::setOffDateOffset (const DateOffset_T & *iDateOffset*) [inline]

Set the off date offset.

Definition at line 82 of file SegmentPeriod.hpp.

References _offDateOffset.

32.140.4.16 void stdair::SegmentPeriod::setElapsedTime (const Duration_T & *iElapsedTime*) [inline]

Set the elapsed time.

Definition at line 87 of file SegmentPeriod.hpp.

References _elapsedTime.

32.140.4.17 void stdair::SegmentPeriod::addCabinBookingClassList (const CabinCode_T & *iCabinCode*, const ClassList_String_T & *iClassCodeList*)

Add a pair cabin code and list of class codes within the cabin to the cabin booking class map.

Definition at line 41 of file SegmentPeriod.cpp.

References _cabinBookingClassMap.

32.140.4.18 void stdair::SegmentPeriod::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 100 of file SegmentPeriod.hpp.

References toString().

32.140.4.19 void stdair::SegmentPeriod::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 104 of file SegmentPeriod.hpp.

32.140.4.20 std::string stdair::SegmentPeriod::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 33 of file SegmentPeriod.cpp.

References describeKey().

Referenced by toStream().

32.140.4.21 const std::string stdair::SegmentPeriod::describeKey () const [inline]

Get a string describing the key.

Definition at line 110 of file SegmentPeriod.hpp.

References _key, and stdair::SegmentPeriodKey::toString().

Referenced by toString().

32.140.5 Friends And Related Function Documentation

32.140.5.1 friend class FacBom [friend]

Definition at line 16 of file SegmentPeriod.hpp.

32.140.5.2 friend class FacCloneBom [friend]

Definition at line 17 of file SegmentPeriod.hpp.

32.140.5.3 friend class FacBomManager [friend]

Definition at line 18 of file SegmentPeriod.hpp.

32.140.6 Member Data Documentation

32.140.6.1 Key_T stdair::SegmentPeriod::_key [protected]

Definition at line 135 of file SegmentPeriod.hpp.

Referenced by describeKey(), getBoardingPoint(), getKey(), and getOffPoint().

32.140.6.2 BomAbstract* stdair::SegmentPeriod::_parent [protected]

Definition at line 136 of file SegmentPeriod.hpp.

Referenced by getParent().

32.140.6.3 Duration_T stdair::SegmentPeriod::_boardingTime [protected]

Definition at line 137 of file SegmentPeriod.hpp.

Referenced by getBoardingTime(), and setBoardingTime().

32.140.6.4 Duration_T stdair::SegmentPeriod::_offTime [protected]

Definition at line 138 of file SegmentPeriod.hpp.

Referenced by getOffTime(), and setOffTime().

32.140.6.5 DateOffset_T stdair::SegmentPeriod::_boardingDateOffset [protected]

Definition at line 139 of file SegmentPeriod.hpp.

Referenced by getBoardingDateOffset(), and setBoardingDateOffset().

32.140.6.6 DateOffset_T stdair::SegmentPeriod::_offDateOffset [protected]

Definition at line 140 of file SegmentPeriod.hpp.

Referenced by getOffDateOffset(), and setOffDateOffset().

32.140.6.7 Duration_T stdair::SegmentPeriod::_elapsedTime [protected]

Definition at line 141 of file SegmentPeriod.hpp.

Referenced by getElapsedTime(), and setElapsedTime().

32.140.6.8 CabinBookingClassMap_T stdair::SegmentPeriod::_cabinBookingClassMap [protected]

Definition at line 142 of file SegmentPeriod.hpp.

Referenced by addCabinBookingClassList(), and getCabinBookingClassMap().

32.140.6.9 HolderMap_T stdair::SegmentPeriod::_holderMap [protected]

Definition at line 143 of file SegmentPeriod.hpp.

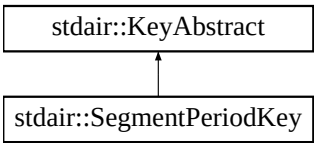
Referenced by getHolderMap().

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

- stdair/bom/SegmentPeriod.hpp
- stdair/bom/SegmentPeriod.cpp

32.141 stdair::SegmentPeriodKey Struct Reference

`#include <stdair/bom/SegmentPeriodKey.hpp>`Inheritance diagram for stdair::SegmentPeriodKey::



Public Member Functions

- SegmentPeriodKey (const AirportCode_T &, const AirportCode_T &)
- SegmentPeriodKey (const SegmentPeriodKey &)
- ~SegmentPeriodKey ()
- const AirportCode_T & getBoardingPoint () const
- const AirportCode_T & getOffPoint () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.141.1 Detailed Description

Key of segment-period.

Definition at line 14 of file SegmentPeriodKey.hpp.

32.141.2 Constructor & Destructor Documentation

32.141.2.1 stdair::SegmentPeriodKey::SegmentPeriodKey (const AirportCode_T & *iBoardingPoint*, const AirportCode_T & *iOffPoint*)

Constructors.

Definition at line 12 of file SegmentPeriodKey.cpp.

32.141.2.2 stdair::SegmentPeriodKey::SegmentPeriodKey (const SegmentPeriodKey & *iKey*)

Definition at line 18 of file SegmentPeriodKey.cpp.

32.141.2.3 stdair::SegmentPeriodKey::~SegmentPeriodKey ()

Destructor.

Definition at line 23 of file SegmentPeriodKey.cpp.

32.141.3 Member Function Documentation

32.141.3.1 const AirportCode_T& stdair::SegmentPeriodKey::getBoardingPoint () const `[inline]`

Get the boardinging point.

Definition at line 29 of file SegmentPeriodKey.hpp.

Referenced by stdair::SegmentPeriod::getBoardingPoint().

32.141.3.2 const AirportCode_T& stdair::SegmentPeriodKey::getOffPoint () const `[inline]`

Get the arrival point.

Definition at line 34 of file SegmentPeriodKey.hpp.

Referenced by stdair::SegmentPeriod::getOffPoint().

32.141.3.3 void stdair::SegmentPeriodKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 27 of file SegmentPeriodKey.cpp.

References toString().

32.141.3.4 void stdair::SegmentPeriodKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 32 of file SegmentPeriodKey.cpp.

32.141.3.5 const std::string stdair::SegmentPeriodKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-period.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file SegmentPeriodKey.cpp.

Referenced by stdair::SegmentPeriod::describeKey(), and toStream().

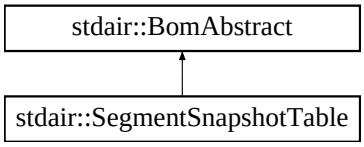
The documentation for this struct was generated from the following files:

- stdair/bom/SegmentPeriodKey.hpp
- stdair/bom/SegmentPeriodKey.cpp

32.142 stdair::SegmentSnapshotTable Class Reference

Class representing the actual attributes for an airline segment data tables.

`#include <stdair/bom/SegmentSnapshotTable.hpp>`Inheritance diagram for stdair::SegmentSnapshotTable::



Public Types

- typedef SegmentSnapshotTableKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const TableID_T & getTableID () const
- const HolderMap_T & getHolderMap () const
- const SegmentCabinIndexMap_T & getSegmentCabinIndexMap () const
- const ClassIndexMap_T & getClassIndexMap () const
- const ClassIndex_T & getClassIndex (const MapKey_T &) const
- const SegmentDataID_T & getSegmentDataID (const SegmentCabin &) const
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const

- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- ConstSegmentCabinDTDSnapshotView_T getConstSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const
- ConstSegmentCabinDTDRangeSnapshotView_T getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const
- SegmentCabinDTDSnapshotView_T getSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T)
- SegmentCabinDTDRangeSnapshotView_T getSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T)
- void initSnapshotBlocks (const SegmentCabinIndexMap_T &, const ClassIndexMap_T &)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Protected Member Functions

- SegmentSnapshotTable (const Key_T &)
- virtual ~SegmentSnapshotTable ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap
- SegmentCabinIndexMap_T _segmentCabinIndexMap
- ClassIndexMap_T _classIndexMap
- SnapshotBlock_T _bookingSnapshotBlock
- SnapshotBlock_T _cancellationSnapshotBlock
- SnapshotBlock_T _productOrientedNetBookingSnapshotBlock
- SnapshotBlock_T _priceOrientedNetBookingSnapshotBlock
- SnapshotBlock_T _productOrientedGrossBookingSnapshotBlock
- SnapshotBlock_T _priceOrientedGrossBookingSnapshotBlock
- SnapshotBlock_T _availabilitySnapshotBlock

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

32.142.1 Detailed Description

Class representing the actual attributes for an airline segment data tables.

Definition at line 31 of file SegmentSnapshotTable.hpp.

32.142.2 Member Typedef Documentation

32.142.2.1 typedef SegmentSnapshotTableKey stdair::SegmentSnapshotTable::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 41 of file SegmentSnapshotTable.hpp.

32.142.3 Constructor & Destructor Documentation

32.142.3.1 stdair::SegmentSnapshotTable::SegmentSnapshotTable (const Key_T & *iKey*) [protected]

Main constructor.

Definition at line 34 of file SegmentSnapshotTable.cpp.

32.142.3.2 stdair::SegmentSnapshotTable::~SegmentSnapshotTable () [protected, virtual]

Destructor.

Definition at line 38 of file SegmentSnapshotTable.cpp.

32.142.4 Member Function Documentation

32.142.4.1 const Key_T& stdair::SegmentSnapshotTable::getKey () const [inline]

Get the segment data table key.

Definition at line 47 of file SegmentSnapshotTable.hpp.

References _key.

32.142.4.2 BomAbstract* const stdair::SegmentSnapshotTable::getParent () const [inline]

Get the parent object.

Definition at line 52 of file SegmentSnapshotTable.hpp.

References _parent.

32.142.4.3 const TableID_T& stdair::SegmentSnapshotTable::getTableID () const [inline]

Get the table ID (part of the primary key).

Definition at line 57 of file SegmentSnapshotTable.hpp.

References _key, and stdair::SegmentSnapshotTableKey::getTableID().

32.142.4.4 const HolderMap_T& stdair::SegmentSnapshotTable::getHolderMap () const [inline]

Get the map of children holders.

Definition at line 64 of file SegmentSnapshotTable.hpp.

References _holderMap.

32.142.4.5 const SegmentCabinIndexMap_T& stdair::SegmentSnapshotTable::getSegmentCabinIndexMap () const [inline]

Get the segment-cabin index map.

Definition at line 69 of file SegmentSnapshotTable.hpp.

References _segmentCabinIndexMap.

32.142.4.6 const ClassIndexMap_T& stdair::SegmentSnapshotTable::getClassIndexMap () const [inline]

Get the class index map.

Definition at line 74 of file SegmentSnapshotTable.hpp.

References _classIndexMap.

32.142.4.7 const ClassIndex_T & stdair::SegmentSnapshotTable::getClassIndex (const MapKey_T & *iKey*) const

Get the index corresponding to the given class.

Definition at line 88 of file SegmentSnapshotTable.cpp.

References _classIndexMap.

32.142.4.8 const SegmentDataID_T & stdair::SegmentSnapshotTable::getSegmentDataID (const SegmentCabin & *iSegmentCabin*) const

Get the segment data ID corresponding to the givent segment-cabin.

Definition at line 97 of file SegmentSnapshotTable.cpp.

References _segmentCabinIndexMap.

32.142.4.9 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 106 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

32.142.4.10 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 119 of file SegmentSnapshotTable.cpp.

32.142.4.11 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 130 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

32.142.4.12 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 142 of file SegmentSnapshotTable.cpp.

References _bookingSnapshotBlock, and _classIndexMap.

32.142.4.13 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 155 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

32.142.4.14 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 168 of file SegmentSnapshotTable.cpp.

32.142.4.15 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 179 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

32.142.4.16 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 191 of file SegmentSnapshotTable.cpp.

References _cancellationSnapshotBlock, and _classIndexMap.

32.142.4.17 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 204 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

32.142.4.18 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshot (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 217 of file SegmentSnapshotTable.cpp.

32.142.4.19 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 228 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

32.142.4.20 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 240 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedNetBookingSnapshotBlock.

32.142.4.21 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 254 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

32.142.4.22 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotVie (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 267 of file SegmentSnapshotTable.cpp.

32.142.4.23 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 278 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

32.142.4.24 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 290 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedNetBookingSnapshotBlock.

32.142.4.25 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 303 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

32.142.4.26 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapsh (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 316 of file SegmentSnapshotTable.cpp.

32.142.4.27 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 327 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

32.142.4.28 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotVie (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 339 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _productOrientedGrossBookingSnapshotBlock.

32.142.4.29 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 353 of file SegmentSnapshotTable.cpp.

References _classIndexMap, and _priceOrientedGrossBookingSnapshotBlock.

32.142.4.30 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotV (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 366 of file SegmentSnapshotTable.cpp.

32.142.4.31 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 378 of file SegmentSnapshotTable.cpp.

32.142.4.32 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 390 of file SegmentSnapshotTable.cpp.

32.142.4.33 ConstSegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*) const

Get the const view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 402 of file SegmentSnapshotTable.cpp.

32.142.4.34 ConstSegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*) const

Get the const view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 414 of file SegmentSnapshotTable.cpp.

32.142.4.35 SegmentCabinDTDSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTD*)

Get the view of snapshots for a given DTD and a range of segment-cabins.

Definition at line 425 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, and _classIndexMap.

32.142.4.36 SegmentCabinDTDRangeSnapshotView_T stdair::SegmentSnapshotTable::getSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T *iSCIdxBegin*, const SegmentDataID_T *iSCIdxEnd*, const DTD_T *iDTDBegin*, const DTD_T *iDTDEnd*)

Get the view of snapshots for a given range of DTD and a range of segment-cabins.

Definition at line 437 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, and _classIndexMap.

32.142.4.37 void stdair::SegmentSnapshotTable::initSnapshotBlocks (const SegmentCabinIndexMap_T & *iSegmentCabinIndexMap*, const ClassIndexMap_T & *iClassIndexMap*)

Set the segment-cabin and value type index maps and initialise the snapshot blocks.

Definition at line 50 of file SegmentSnapshotTable.cpp.

References _availabilitySnapshotBlock, _bookingSnapshotBlock, _cancellationSnapshotBlock, _classIndexMap, _priceOrientedGrossBookingSnapshotBlock, _priceOrientedNetBookingSnapshotBlock, _productOrientedGrossBookingSnapshotBlock, _productOrientedNetBookingSnapshotBlock, _segmentCabinIndexMap, and stdair::DEFAULT_MAX_DTD.

32.142.4.38 void stdair::SegmentSnapshotTable::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 283 of file SegmentSnapshotTable.hpp.

References toString().

32.142.4.39 void stdair::SegmentSnapshotTable::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 292 of file SegmentSnapshotTable.hpp.

32.142.4.40 std::string stdair::SegmentSnapshotTable::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 42 of file SegmentSnapshotTable.cpp.

References describeKey().

Referenced by toStream().

32.142.4.41 const std::string stdair::SegmentSnapshotTable::describeKey () const [inline]

Get a string describing the key.

Definition at line 303 of file SegmentSnapshotTable.hpp.

References _key, and stdair::SegmentSnapshotTableKey::toString().

Referenced by toString().

32.142.4.42 template<class Archive > void stdair::SegmentSnapshotTable::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 464 of file SegmentSnapshotTable.cpp.

References _key.

32.142.5 Friends And Related Function Documentation

32.142.5.1 friend class FacBom [friend]

Definition at line 32 of file SegmentSnapshotTable.hpp.

32.142.5.2 friend class FacBomManager [friend]

Definition at line 33 of file SegmentSnapshotTable.hpp.

32.142.5.3 friend class boost::serialization::access [friend]

Definition at line 34 of file SegmentSnapshotTable.hpp.

32.142.6 Member Data Documentation

32.142.6.1 Key_T stdair::SegmentSnapshotTable::_key [protected]

Primary key (table ID and departure block).

Definition at line 352 of file SegmentSnapshotTable.hpp.

Referenced by describeKey(), getKey(), getTableID(), and serialize().

32.142.6.2 BomAbstract* stdair::SegmentSnapshotTable::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 355 of file SegmentSnapshotTable.hpp.

Referenced by getParent().

32.142.6.3 HolderMap_T stdair::SegmentSnapshotTable::_holderMap [protected]

Map holding the children.

Definition at line 358 of file SegmentSnapshotTable.hpp.

Referenced by getHolderMap().

32.142.6.4 SegmentCabinIndexMap_T stdair::SegmentSnapshotTable::_segmentCabinIndexMap [protected]

Map holding the segment-cabin position within the snapshot blocks.

Definition at line 361 of file SegmentSnapshotTable.hpp.

Referenced by getSegmentCabinIndexMap(), getSegmentDataID(), and initSnapshotBlocks().

32.142.6.5 ClassIndexMap_T stdair::SegmentSnapshotTable::_classIndexMap [protected]

Map holding the value type (class, etc) within a a segment-cabin inside the snapshot blocks.

Definition at line 365 of file SegmentSnapshotTable.hpp.

Referenced by getClassIndex(), getClassIndexMap(), getConstSegmentCabinDTDBookingSnapshotView(), getConstSegmentCabinDTDCancellationSnapshotView(), getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView(), getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDAvailabilitySnapshotView(), getSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeAvailabilitySnapshotView(), getSegmentCabinDTDRangeBook-

ingSnapshotView(), getSegmentCabinDTDRangeCancellationSnapshotView(), getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.6 SnapshotBlock_T stdair::SegmentSnapshotTable::_bookingSnapshotBlock [protected]

Booking snapshot block.

Definition at line 368 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDBookingSnapshotView(), getSegmentCabinDTDRangeBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.7 SnapshotBlock_T stdair::SegmentSnapshotTable::_cancellationSnapshotBlock [protected]

Cancellation snapshot block.

Definition at line 371 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDCancellationSnapshotView(), getSegmentCabinDTDRangeCancellationSnapshotView(), and initSnapshotBlocks().

32.142.6.8 SnapshotBlock_T stdair::SegmentSnapshotTable::_productOrientedNetBookingSnapshotBlock [protected]

Product oriented net booking block.

Definition at line 374 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDProductOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.9 SnapshotBlock_T stdair::SegmentSnapshotTable::_priceOrientedNetBookingSnapshotBlock [protected]

Price oriented net booking block.

Definition at line 377 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDPriceOrientedNetBookingSnapshotView(), getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.10 SnapshotBlock_T stdair::SegmentSnapshotTable::_productOrientedGrossBookingSnapshotBlock [protected]

Product oriented gross booking block.

Definition at line 380 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDProductOrientedGrossBookingSnapshotView(), getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.11 SnapshotBlock_T stdair::SegmentSnapshotTable::_-priceOrientedGrossBookingSnapshotBlock [protected]

Price oriented gross booking block.

Definition at line 383 of file SegmentSnapshotTable.hpp.

Referenced by getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView(), and initSnapshotBlocks().

32.142.6.12 SnapshotBlock_T stdair::SegmentSnapshotTable::_availabilitySnapshotBlock [protected]

Availability block.

Definition at line 386 of file SegmentSnapshotTable.hpp.

Referenced by getSegmentCabinDTDAvailabilitySnapshotView(), getSegmentCabinDTDRangeAvailabilitySnapshotView(), and initSnapshotBlocks().

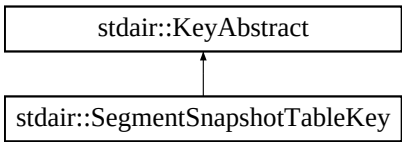
The documentation for this class was generated from the following files:

- stdair/bom/SegmentSnapshotTable.hpp
- stdair/bom/SegmentSnapshotTable.cpp

32.143 stdair::SegmentSnapshotTableKey Struct Reference

Key of a given guillotine block, made of a guillotine number.

`#include <stdair/bom/SegmentSnapshotTableKey.hpp>`Inheritance diagram for stdair::SegmentSnapshotTableKey::



Public Member Functions

- SegmentSnapshotTableKey (const TableID_T &)
- SegmentSnapshotTableKey (const SegmentSnapshotTableKey &)
- ~SegmentSnapshotTableKey ()
- const TableID_T & getTableID () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)

Friends

- class boost::serialization::access

32.143.1 Detailed Description

Key of a given guillotine block, made of a guillotine number.

Definition at line 26 of file SegmentSnapshotTableKey.hpp.

32.143.2 Constructor & Destructor Documentation

32.143.2.1 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey (const TableID_T & *iTableID*)

Constructor.

Definition at line 26 of file SegmentSnapshotTableKey.cpp.

32.143.2.2 stdair::SegmentSnapshotTableKey::SegmentSnapshotTableKey (const SegmentSnapshotTableKey & *iKey*)

Copy constructor.

Definition at line 31 of file SegmentSnapshotTableKey.cpp.

32.143.2.3 stdair::SegmentSnapshotTableKey::∼SegmentSnapshotTableKey ()

Destructor.

Definition at line 36 of file SegmentSnapshotTableKey.cpp.

32.143.3 Member Function Documentation

32.143.3.1 const TableID_T& stdair::SegmentSnapshotTableKey::getTableID () const `[inline]`

Get the table ID.

Definition at line 56 of file SegmentSnapshotTableKey.hpp.

Referenced by stdair::SegmentSnapshotTable::getTableID().

32.143.3.2 void stdair::SegmentSnapshotTableKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 40 of file SegmentSnapshotTableKey.cpp.

References toString().

32.143.3.3 void stdair::SegmentSnapshotTableKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file SegmentSnapshotTableKey.cpp.

32.143.3.4 const std::string stdair::SegmentSnapshotTableKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-block.

Reimplemented from stdair::KeyAbstract.

Definition at line 49 of file SegmentSnapshotTableKey.cpp.

Referenced by stdair::SegmentSnapshotTable::describeKey(), and toStream().

32.143.3.5 template<class Archive > void stdair::SegmentSnapshotTableKey::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

Definition at line 71 of file SegmentSnapshotTableKey.cpp.

32.143.4 Friends And Related Function Documentation

32.143.4.1 friend class boost::serialization::access [friend]

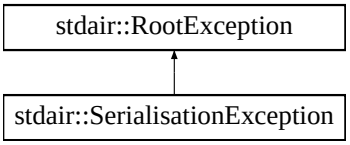
Definition at line 27 of file SegmentSnapshotTableKey.hpp.

The documentation for this struct was generated from the following files:

- stdair/bom/SegmentSnapshotTableKey.hpp
- stdair/bom/SegmentSnapshotTableKey.cpp

32.144 stdair::SerialisationException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::SerialisationException::



Public Member Functions

- SerialisationException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.144.1 Detailed Description

Serialisation.

Definition at line 119 of file stdair_exceptions.hpp.

32.144.2 Constructor & Destructor Documentation

32.144.2.1 stdair::SerialisationException::SerialisationException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 122 of file stdair_exceptions.hpp.

32.144.3 Member Function Documentation

32.144.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.144.4 Member Data Documentation

32.144.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

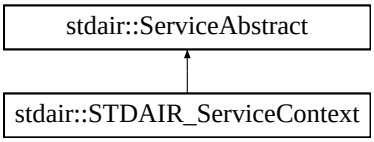
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.145 stdair::ServiceAbstract Class Reference

`#include <stdair/service/ServiceAbstract.hpp>`Inheritance diagram for stdair::ServiceAbstract::



Public Member Functions

- virtual ~ServiceAbstract ()
- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Protected Member Functions

- ServiceAbstract ()

32.145.1 Detailed Description

Base class for the Service layer.

Definition at line 15 of file ServiceAbstract.hpp.

32.145.2 Constructor & Destructor Documentation

32.145.2.1 virtual stdair::ServiceAbstract::~ServiceAbstract () `[inline, virtual]`

Destructor.

Definition at line 21 of file ServiceAbstract.hpp.

32.145.2.2 stdair::ServiceAbstract::ServiceAbstract () `[inline, protected]`

Display of the structure. Protected Default Constructor to ensure this class is abtract.

Definition at line 46 of file ServiceAbstract.hpp.

32.145.3 Member Function Documentation

32.145.3.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 28 of file ServiceAbstract.hpp.

32.145.3.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 35 of file ServiceAbstract.hpp.

Referenced by operator>>().

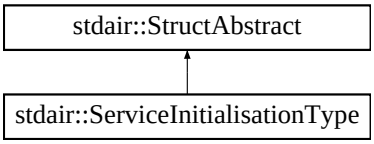
The documentation for this class was generated from the following file:

- stdair/service/ServiceAbstract.hpp

32.146 stdair::ServiceInitialisationType Struct Reference

Enumeration of service initialisation types.

`#include <stdair/basic/ServiceInitialisationType.hpp>`Inheritance diagram for stdair::ServiceInitialisationType::



Public Types

- enum EN_ServiceInitialisationType { NOT_YET_INITIALISED = 0, FILE_PARSING, BUILTIN_SAMPLE, LAST_VALUE }

Public Member Functions

- EN_ServiceInitialisationType getType () const
- char getTypeAsChar () const

- std::string getTypeAsString () const
- const std::string describe () const
- bool operator== (const EN_ServiceInitialisationType &) const
- ServiceInitialisationType (const EN_ServiceInitialisationType &)
- ServiceInitialisationType (const char iType)
- ServiceInitialisationType (const std::string &iType)
- ServiceInitialisationType (const ServiceInitialisationType &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_ServiceInitialisationType &)
- static EN_ServiceInitialisationType getType (const char)
- static char getTypeLabel (const EN_ServiceInitialisationType &)
- static std::string getTypeLabelAsString (const EN_ServiceInitialisationType &)
- static std::string describeLabels ()

32.146.1 Detailed Description

Enumeration of service initialisation types.

Definition at line 17 of file ServiceInitialisationType.hpp.

32.146.2 Member Enumeration Documentation

32.146.2.1 enum stdair::ServiceInitialisationType::EN_ServiceInitialisationType

Enumerator:

NOT_YET_INITIALISED

FILE_PARSING

BUILTIN_SAMPLE

LAST_VALUE

Definition at line 19 of file ServiceInitialisationType.hpp.

32.146.3 Constructor & Destructor Documentation

32.146.3.1 stdair::ServiceInitialisationType::ServiceInitialisationType (const EN_ServiceInitialisationType & *iServiceInitialisationType*)

Main constructor.

Definition at line 36 of file ServiceInitialisationType.cpp.

32.146.3.2 stdair::ServiceInitialisationType::ServiceInitialisationType (const char *iType*)

Alternative constructor.

Definition at line 65 of file ServiceInitialisationType.cpp.

32.146.3.3 stdair::ServiceInitialisationType::ServiceInitialisationType (const std::string & *iType*)

Alternative constructor.

Definition at line 71 of file ServiceInitialisationType.cpp.

References getType().

32.146.3.4 stdair::ServiceInitialisationType::ServiceInitialisationType (const ServiceInitialisationType & *iServiceInitialisationType*)

Default copy constructor.

Definition at line 30 of file ServiceInitialisationType.cpp.

32.146.4 Member Function Documentation

32.146.4.1 const std::string & stdair::ServiceInitialisationType::getLabel (const EN_ServiceInitialisationType & *iType*) `[static]`

Get the label as a string (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Definition at line 81 of file ServiceInitialisationType.cpp.

32.146.4.2 ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType (const char *iTypeChar*) `[static]`

Get the type value from parsing a single char (e.g., 'N', 'F', 'B').

Definition at line 42 of file ServiceInitialisationType.cpp.

References BUILTIN_SAMPLE, describeLabels(), FILE_PARSING, LAST_VALUE, and NOT_YET_-INITIALISED.

32.146.4.3 char stdair::ServiceInitialisationType::getTypeLabel (const EN_ServiceInitialisationType & *iType*) `[static]`

Get the label as a single char (e.g., 'N', 'F', 'B').

Definition at line 87 of file ServiceInitialisationType.cpp.

32.146.4.4 std::string stdair::ServiceInitialisationType::getTypeLabelAsString (const EN_ServiceInitialisationType & *iType*) `[static]`

Get the label as a string of a single char (e.g., "N", "F", "B").

Definition at line 93 of file ServiceInitialisationType.cpp.

32.146.4.5 std::string stdair::ServiceInitialisationType::describeLabels () [static]

List the labels.

Definition at line 100 of file ServiceInitialisationType.cpp.

References LAST_VALUE.

Referenced by getType().

32.146.4.6 ServiceInitialisationType::EN_ServiceInitialisationType stdair::ServiceInitialisationType::getType () const

Get the enumerated value.

Definition at line 113 of file ServiceInitialisationType.cpp.

Referenced by ServiceInitialisationType().

32.146.4.7 char stdair::ServiceInitialisationType::getTypeAsChar () const

Get the enumerated value as a short string (e.g., 'N', 'F', 'B').

Definition at line 118 of file ServiceInitialisationType.cpp.

32.146.4.8 std::string stdair::ServiceInitialisationType::getTypeAsString () const

Get the enumerated value as a short string (e.g., "N", "F", "B").

Definition at line 124 of file ServiceInitialisationType.cpp.

32.146.4.9 const std::string stdair::ServiceInitialisationType::describe () const [virtual]

Give a description of the structure (e.g., "Not yet initialised", "File parsing" or "Built-in sample BOM").

Implements stdair::StructAbstract.

Definition at line 131 of file ServiceInitialisationType.cpp.

32.146.4.10 bool stdair::ServiceInitialisationType::operator== (const EN_ServiceInitialisationType & *iType*) const

Comparison operator.

Definition at line 139 of file ServiceInitialisationType.cpp.

32.146.4.11 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline, inherited]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.146.4.12 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

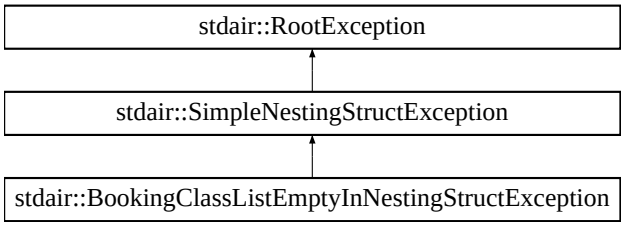
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/ServiceInitialisationType.hpp
- stdair/basic/ServiceInitialisationType.cpp

32.147 stdair::SimpleNestingStructException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::SimpleNestingStructException::



Public Member Functions

- SimpleNestingStructException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.147.1 Detailed Description

Simple Nesting Structure.

Definition at line 211 of file stdair_exceptions.hpp.

32.147.2 Constructor & Destructor Documentation

32.147.2.1 stdair::SimpleNestingStructException::SimpleNestingStructException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 214 of file stdair_exceptions.hpp.

32.147.3 Member Function Documentation

32.147.3.1 const char* stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.147.4 Member Data Documentation

32.147.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

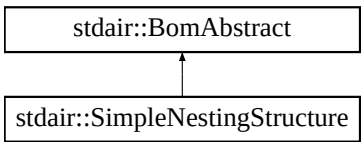
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.148 stdair::SimpleNestingStructure Class Reference

`#include <stdair/bom/SimpleNestingStructure.hpp>`Inheritance diagram for stdair::SimpleNestingStructure::



Public Types

- typedef NestingStructureKey Key_T

Public Member Functions

- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const NestingNodeList_T & getNestingNodeList () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- template<class Archive >
  void serialize (Archive &ar, const unsigned int iFileVersion)
- SimpleNestingStructure (const Key_T &)
- virtual ~SimpleNestingStructure ()

Friends

- class FacBom
- class FacBomManager
- class boost::serialization::access

32.148.1 Detailed Description

Structure holding a nesting node map according to the yield.

Definition at line 26 of file SimpleNestingStructure.hpp.

32.148.2 Member Typedef Documentation

32.148.2.1 typedef NestingStructureKey stdair::SimpleNestingStructure::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 36 of file SimpleNestingStructure.hpp.

32.148.3 Constructor & Destructor Documentation

32.148.3.1 stdair::SimpleNestingStructure::SimpleNestingStructure (const Key_T & *iKey*)

Main constructor.

Definition at line 36 of file SimpleNestingStructure.cpp.

32.148.3.2 stdair::SimpleNestingStructure::~SimpleNestingStructure () `[virtual]`

Destructor.

Definition at line 41 of file SimpleNestingStructure.cpp.

32.148.4 Member Function Documentation

32.148.4.1 const Key_T& stdair::SimpleNestingStructure::getKey () const `[inline]`

Get the nesting key.

Definition at line 41 of file SimpleNestingStructure.hpp.

32.148.4.2 BomAbstract* const stdair::SimpleNestingStructure::getParent () const `[inline]`

Get the parent object.

Definition at line 46 of file SimpleNestingStructure.hpp.

32.148.4.3 const HolderMap_T& stdair::SimpleNestingStructure::getHolderMap () const `[inline]`

Get the map of children holders.

Definition at line 53 of file SimpleNestingStructure.hpp.

32.148.4.4 const NestingNodeList_T & stdair::SimpleNestingStructure::getNestingNodeList () const

Get the nesting node list

Definition at line 115 of file SimpleNestingStructure.cpp.

32.148.4.5 void stdair::SimpleNestingStructure::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 69 of file SimpleNestingStructure.hpp.

References toString().

32.148.4.6 void stdair::SimpleNestingStructure::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 78 of file SimpleNestingStructure.hpp.

32.148.4.7 std::string stdair::SimpleNestingStructure::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 107 of file SimpleNestingStructure.cpp.

References describeKey().

Referenced by toStream().

32.148.4.8 const std::string stdair::SimpleNestingStructure::describeKey () const [inline]

Get a string describing the key.

Definition at line 89 of file SimpleNestingStructure.hpp.

References stdair::NestingStructureKey::toString().

Referenced by toString().

32.148.4.9 template<class Archive > void stdair::SimpleNestingStructure::serialize (Archive & *ar*, const unsigned int *iFileVersion*) [inline]

Serialisation.

32.148.5 Friends And Related Function Documentation

32.148.5.1 friend class FacBom [friend]

Definition at line 27 of file SimpleNestingStructure.hpp.

32.148.5.2 friend class FacBomManager [friend]

Definition at line 28 of file SimpleNestingStructure.hpp.

32.148.5.3 friend class boost::serialization::access [friend]

Definition at line 29 of file SimpleNestingStructure.hpp.

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

- stdair/bom/SimpleNestingStructure.hpp
- stdair/bom/SimpleNestingStructure.cpp

32.149 swift::SKeymap Class Reference

The readline keymap wrapper.

`#include <stdair/ui/cmdline/SReadline.hpp>`

Public Member Functions

- SKeymap (bool PrintableBound=false)
  Creates a new keymap.
- SKeymap (Keymap Pattern)
  Creates a new keymap which is a copy of Pattern.
- ~SKeymap ()
  Frees the allocated keymap.
- void Bind (int Key, KeyCallback Callback)
  Binds the given key to a function.
- void Unbind (int Key)
  Unbinds the given key.
- SKeymap (const SKeymap &rhs)
  Copy constructor.
- SKeymap & operator= (const SKeymap &rhs)
  operator=

Friends

- class SReadline

32.149.1 Detailed Description

The readline keymap wrapper. Attention: It is not thread safe! Supports: key binding, key unbinding

Definition at line 307 of file SReadline.hpp.

32.149.2 Constructor & Destructor Documentation

32.149.2.1 swift::SKeymap::SKeymap (bool *PrintableBound* = `false`) `[inline, explicit]`

Creates a new keymap.

Parameters:

PrintableBound if true - the printable characters are bound if false - the keymap is empty

Definition at line 319 of file SReadline.hpp.

32.149.2.2 swift::SKeymap::SKeymap (Keymap *Pattern*) `[inline, explicit]`

Creates a new keymap which is a copy of Pattern.

Parameters:

Pattern A keymap to be copied.

Definition at line 342 of file SReadline.hpp.

32.149.2.3 swift::SKeymap::∼SKeymap () `[inline]`

Frees the allocated keymap.

Definition at line 354 of file SReadline.hpp.

32.149.2.4 swift::SKeymap::SKeymap (const SKeymap & *rhs*) `[inline]`

Copy constructor.

Parameters:

rhs Right hand side object of SKeymap

Definition at line 395 of file SReadline.hpp.

32.149.3 Member Function Documentation

32.149.3.1 void swift::SKeymap::Bind (int *Key*, KeyCallback *Callback*) `[inline]`

Binds the given key to a function.

Parameters:

Key A key to be bound

Callback A function to be called when the Key is pressed

Definition at line 366 of file SReadline.hpp.

32.149.3.2 void swift::SKeymap::Unbind (int *Key*) `[inline]`

Unbinds the given key.

Parameters:

Key A key to be unbound

Definition at line 381 of file SReadline.hpp.

32.149.3.3 SKeymap& swift::SKeymap::operator= (const SKeymap & *rhs*) `[inline]`

operator=

Parameters:

rhs Right hand side object of SKeymap

Definition at line 407 of file SReadline.hpp.

32.149.4 Friends And Related Function Documentation

32.149.4.1 friend class SReadline `[friend]`

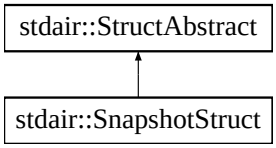
Definition at line 415 of file SReadline.hpp.

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

- stdair/ui/cmdline/SReadline.hpp

32.150 stdair::SnapshotStruct Struct Reference

`#include <stdair/bom/SnapshotStruct.hpp>`Inheritance diagram for stdair::SnapshotStruct::



Public Member Functions

- const AirlineCode_T & getAirlineCode () const
- const DateTime_T & getSnapshotTime () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- SnapshotStruct (const AirlineCode_T &, const DateTime_T &)
- SnapshotStruct (const SnapshotStruct &)
- ~SnapshotStruct ()

32.150.1 Detailed Description

Structure holding the elements of a snapshot .

Definition at line 19 of file SnapshotStruct.hpp.

32.150.2 Constructor & Destructor Documentation

32.150.2.1 stdair::SnapshotStruct::SnapshotStruct (const AirlineCode_T & *iAirlineCode*, const DateTime_T & *iSnapshotTime*)

Constructor.

Definition at line 26 of file SnapshotStruct.cpp.

32.150.2.2 stdair::SnapshotStruct::SnapshotStruct (const SnapshotStruct & *iSnapshot*)

Copy constructor.

Definition at line 19 of file SnapshotStruct.cpp.

32.150.2.3 stdair::SnapshotStruct::~SnapshotStruct ()

Destructor.

Definition at line 32 of file SnapshotStruct.cpp.

32.150.3 Member Function Documentation

32.150.3.1 const AirlineCode_T& stdair::SnapshotStruct::getAirlineCode () const `[inline]`

Get the airline code.

Definition at line 23 of file SnapshotStruct.hpp.

32.150.3.2 const DateTime_T& stdair::SnapshotStruct::getSnapshotTime () const `[inline]`

Get the snapshot action time.

Definition at line 28 of file SnapshotStruct.hpp.

32.150.3.3 void stdair::SnapshotStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file SnapshotStruct.cpp.

References describe().

32.150.3.4 void stdair::SnapshotStruct::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file SnapshotStruct.cpp.

32.150.3.5 const std::string stdair::SnapshotStruct::describe () const `[virtual]`

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file SnapshotStruct.cpp.

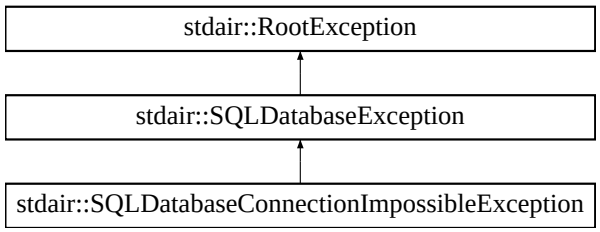
Referenced by toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/SnapshotStruct.hpp
- stdair/bom/SnapshotStruct.cpp

32.151 stdair::SQLDatabaseConnectionImpossibleException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::SQLDatabaseConnectionImpossibleException::



Public Member Functions

- SQLDatabaseConnectionImpossibleException (const std::string &iWhat)
- const char ∗ what () const throw ()

Protected Attributes

- std::string _what

32.151.1 Detailed Description

Database connection.

Definition at line 196 of file stdair_exceptions.hpp.

32.151.2 Constructor & Destructor Documentation

32.151.2.1 stdair::SQLDatabaseConnectionImpossibleException::SQLDatabaseConnectionImpossibleException (const std::string & *iWhat*) `[inline]`

Constructor.

Definition at line 199 of file stdair_exceptions.hpp.

32.151.3 Member Function Documentation

32.151.3.1 const char∗ stdair::RootException::what () const throw () `[inline, inherited]`

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.151.4 Member Data Documentation

32.151.4.1 std::string stdair::RootException::_what `[protected, inherited]`

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

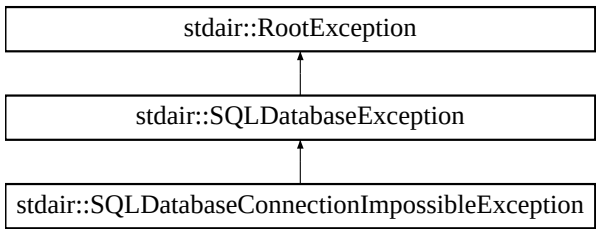
Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.152 stdair::SQLDatabaseException Class Reference

`#include <stdair/stdair_exceptions.hpp>`Inheritance diagram for stdair::SQLDatabaseException::



Public Member Functions

- SQLDatabaseException (const std::string &iWhat)
- const char * what () const throw ()

Protected Attributes

- std::string _what

32.152.1 Detailed Description

Database.

Definition at line 181 of file stdair_exceptions.hpp.

32.152.2 Constructor & Destructor Documentation

32.152.2.1 stdair::SQLDatabaseException::SQLDatabaseException (const std::string & *iWhat*) [inline]

Constructor.

Definition at line 184 of file stdair_exceptions.hpp.

32.152.3 Member Function Documentation

32.152.3.1 const char* stdair::RootException::what () const throw () [inline, inherited]

Give the details of the exception.

Definition at line 38 of file stdair_exceptions.hpp.

References stdair::RootException::_what.

Referenced by stdair::ConfigHolderStruct::updateAirlineFeatures().

32.152.4 Member Data Documentation

32.152.4.1 std::string stdair::RootException::_what [protected, inherited]

Details for the exception.

Definition at line 46 of file stdair_exceptions.hpp.

Referenced by stdair::RootException::what().

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

- stdair/stdair_exceptions.hpp

32.153 swift::SReadline Class Reference

The readline library wrapper.

`#include <stdair/ui/cmdline/SReadline.hpp>`

Public Member Functions

- SReadline (const size_t Limit=DefaultHistoryLimit)

  Constructs the object, sets the completion function.

- SReadline (const std::string &historyFileName, const size_t Limit=DefaultHistoryLimit)

  Constructs the object, sets the completion function, loads history.

- ~SReadline ()

  Saves the session history (if the file name was provided) and destroys the object.

- std::string GetLine (const std::string &Prompt)

  Gets a single line from a user.

- template<typename Container >
  std::string GetLine (const std::string &Prompt, Container &ReadTokens)

  Gets a single line from a user.

- template<typename Container >
  std::string GetLine (const std::string &Prompt, Container &ReadTokens, bool &BreakOut)

  Gets a single line from a user.

- std::string GetLine (const std::string &Prompt, bool &BreakOut)

  Gets a single line from a user.

- template<typename ContainerType >
  void GetHistory (ContainerType &Container)

  Fills the given container with the current history list.

- bool SaveHistory (std::ostream &OS)

  Saves the history to the given file stream.

- bool SaveHistory (const std::string &FileName)

  Saves the history to the given file.

- void ClearHistory ()

  Clears the history. Does not affect the file where the previous session history is saved.

- bool LoadHistory (std::istream &IS)

Loads a history from a file stream.

- bool LoadHistory (const std::string &FileName)

  Loads a history from the given file.

- template<typename ContainerType >
  void RegisterCompletions (const ContainerType &Container)

  Allows to register custom completers.

- void SetKeymap (SKeymap &NewKeymap)

  Sets the given keymap.

32.153.1 Detailed Description

The readline library wrapper. Attention: It is not thread safe! Supports: editing, history, custom completers

Definition at line 424 of file SReadline.hpp.

32.153.2 Constructor & Destructor Documentation

32.153.2.1 swift::SReadline::SReadline (const size_t *Limit* = `DefaultHistoryLimit`) `[inline]`

Constructs the object, sets the completion function.

Parameters:

Limit History size

Definition at line 431 of file SReadline.hpp.

32.153.2.2 swift::SReadline::SReadline (const std::string & *historyFileName*, const size_t *Limit* = `DefaultHistoryLimit`) `[inline]`

Constructs the object, sets the completion function, loads history.

Parameters:

historyFileName File name to load history from

Limit History size

Definition at line 446 of file SReadline.hpp.

References LoadHistory().

32.153.2.3 swift::SReadline::~SReadline () [inline]

Saves the session history (if the file name was provided) and destroys the object.

Definition at line 462 of file SReadline.hpp.

References SaveHistory().

32.153.3 Member Function Documentation

32.153.3.1 std::string swift::SReadline::GetLine (const std::string & *Prompt*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

Returns:

A string which was actually inputed

Definition at line 473 of file SReadline.hpp.

Referenced by GetLine().

32.153.3.2 template<typename Container > std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

ReadTokens A user inputed string splitted into tokens. The container is cleared first

Returns:

A string which was actually inputed

Definition at line 487 of file SReadline.hpp.

References GetLine().

32.153.3.3 template<typename Container > std::string swift::SReadline::GetLine (const std::string & *Prompt*, Container & *ReadTokens*, bool & *BreakOut*) [inline]

Gets a single line from a user.

Parameters:

Prompt A printed prompt

BreakOut it is set to true if the EOF found

ReadTokens A user inputed string splitted into tokens. The container is cleared first

Returns:

A string which was actually inputed

Definition at line 502 of file SReadline.hpp.

References GetLine().

32.153.3.4 std::string swift::SReadline::GetLine (const std::string & *Prompt*, bool & *BreakOut*) `[inline]`

Gets a single line from a user.

Parameters:

Prompt A printed prompt

BreakOut it is set to true if the EOF found

Returns:

A string which was actually inputed

Definition at line 517 of file SReadline.hpp.

32.153.3.5 template<typename ContainerType > void swift::SReadline::GetHistory (ContainerType & *Container*) `[inline]`

Fills the given container with the current history list. Does not clear the given container

Definition at line 552 of file SReadline.hpp.

32.153.3.6 bool swift::SReadline::SaveHistory (std::ostream & *OS*) `[inline]`

Saves the history to the given file stream.

Parameters:

OS output file stream

Returns:

true if success

Definition at line 564 of file SReadline.hpp.

Referenced by SaveHistory(), and ~SReadline().

32.153.3.7 bool swift::SReadline::SaveHistory (const std::string & *FileName*) [inline]

Saves the history to the given file.

Parameters:

FileName File name to save the history to

Returns:

true if success

Definition at line 581 of file SReadline.hpp.

References SaveHistory().

32.153.3.8 void swift::SReadline::ClearHistory () [inline]

Clears the history. Does not affect the file where the previous session history is saved.

Definition at line 594 of file SReadline.hpp.

Referenced by LoadHistory().

32.153.3.9 bool swift::SReadline::LoadHistory (std::istream & *IS*) [inline]

Loads a history from a file stream.

Parameters:

IS Input file stream

Returns:

true if success

Definition at line 604 of file SReadline.hpp.

References ClearHistory().

Referenced by LoadHistory(), and SReadline().

32.153.3.10 bool swift::SReadline::LoadHistory (const std::string & *FileName*) [inline]

Loads a history from the given file.

Parameters:

FileName File name to be load from

Returns:

true if success

Definition at line 629 of file SReadline.hpp.

References LoadHistory().

32.153.3.11 template<typename ContainerType > void swift::SReadline::RegisterCompletions (const ContainerType & *Container*) [inline]

Allows to register custom completers. Supports a special keyword: file. It means to use the standard file name completer.

For example the given container elements could be as follows:

- command1 opt1
- command1 opt2 file
- command2
- command2 opt1

Each container element must describe a single possible command line. The container element must have a conversion to std::string operator.

Parameters:

Container A container which has all the user possible commands.

Definition at line 658 of file SReadline.hpp.

32.153.3.12 void swift::SReadline::SetKeymap (SKeymap & *NewKeymap*) [inline]

Sets the given keymap.

Parameters:

NewKeymap The keymap that should be used from now.

Definition at line 675 of file SReadline.hpp.

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

- stdair/ui/cmdline/SReadline.hpp

32.154 stdair::STDAIR_Service Class Reference

Interface for the STDAIR Services.

```
#include <stdair/STDAIR_Service.hpp>
```

Public Member Functions

- STDAIR_Service ()

  Default constructor.

- STDAIR_Service (const BasLogParams &)

  Constructor.

- STDAIR_Service (const BasLogParams &, const BasDBParams &)

  Constructor.

- ~STDAIR_Service ()

  Destructor.

- void buildSampleBom ()
- void buildDummyInventory (const CabinCapacity_T &iCabinCapacity)
- void buildDummyLegSegmentAccesses (BomRoot &)
- void buildSampleTravelSolutionForPricing (TravelSolutionList_T &)
- void buildSampleTravelSolutions (TravelSolutionList_T &)
- BookingRequestStruct buildSampleBookingRequest (const bool isForCRS=false)
- void clonePersistentBom ()

  Clone the persistent Bom.

- std::string jsonExportFlightDateList (const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0) const
- std::string jsonExportFlightDateObjects (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- std::string jsonExportEventObject (const EventStruct &) const
- std::string jsonExportConfiguration () const
- bool jsonImportConfiguration (const JSONString &) const
- std::string list (const AirlineCode_T &iAirlineCode="all", const FlightNumber_T &iFlightNumber=0) const
- std::string listAirportPairDateRange () const
- bool check (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- bool check (const AirportCode_T &, const AirportCode_T &, const Date_T &iDepartureDate) const
- std::string configDisplay () const
- std::string csvDisplay () const
- std::string csvDisplay (const BomRoot &) const
- std::string csvDisplay (const AirlineCode_T &, const FlightNumber_T &, const Date_T &iDepartureDate) const
- std::string csvDisplay (const TravelSolutionList_T &) const
- std::string csvDisplay (const AirportCode_T &, const AirportCode_T &, const Date_T &iDepartureDate) const
- BomRoot & getBomRoot () const

  Get a reference on the BomRoot object.

- BomRoot & getPersistentBomRoot () const

  Get a reference on the BomRoot object.

- BasLogParams getLogParams () const

- const BasDBParams & getDBParams () const
- const ServiceInitialisationType & getServiceInitialisationType () const
- void importINIConfig (const ConfigINIFile &)

  Import the configuration INI input file (format cfg).

- void importConfigValue (const std::string &iValue, const std::string &iPath)
- template<typename ValueType >
  bool exportConfigValue (ValueType &ioValue, const std::string &iPath)
- void updateAirlineFeatures ()

  Update the airline features objects thanks to the configuration holder.

32.154.1 Detailed Description

Interface for the STDAIR Services.

Definition at line 44 of file STDAIR_Service.hpp.

32.154.2 Constructor & Destructor Documentation

32.154.2.1 stdair::STDAIR_Service::STDAIR_Service ()

Default constructor.

Definition at line 45 of file STDAIR_Service.cpp.

32.154.2.2 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams & *iLogParams*)

Constructor. The init() method is called; see the corresponding documentation for more details.

Moreover, a reference on an output stream is given, so that log outputs can be directed onto that stream.

Parameters:

← ***const*** BasLogParams& Parameters for the output log stream.

Definition at line 61 of file STDAIR_Service.cpp.

32.154.2.3 stdair::STDAIR_Service::STDAIR_Service (const BasLogParams & *iLogParams*, const BasDBParams & *iDBParams*)

Constructor. The init() method is called; see the corresponding documentation for more details.

A reference on an output stream is given, so that log outputs can be directed onto that stream.

Moreover, database connection parameters are given, so that database events can use the corresponding access.

Parameters:

← ***const*** BasLogParams& Parameters for the output log stream.

← ***const*** BasDBParams& Parameters for the database session.

Definition at line 75 of file STDAIR_Service.cpp.

32.154.2.4 stdair::STDAIR_Service::~STDAIR_Service ()

Destructor.

Definition at line 93 of file STDAIR_Service.cpp.

32.154.3 Member Function Documentation

32.154.3.1 void stdair::STDAIR_Service::buildSampleBom ()

Build a sample BOM tree, and attach it to the BomRoot instance.

As for now, a single sample BOM tree is built, with objects for all the simulator-related components, i.e.:

- schedule (e.g., AirSched),
- inventory (e.g., AirInv),
- revenue management (e.g., RMOL),
- pricing (e.g., SimFQT),
- revenue accounting (e.g., AirRAC),
- demand generation (e.g., TraDemGen),
- customer choice (e.g., TravelCCM),
- event manager (e.g., SEvMgr)

Most of the inventories just contain one flight. One of those flights has two legs (and therefore three segments).

Definition at line 172 of file STDAIR_Service.cpp.

32.154.3.2 void stdair::STDAIR_Service::buildDummyInventory (const CabinCapacity_T & *iCabinCapacity*)

Build a dummy inventory, containing a dummy flight-date with a single leg-cabin and some virtual booking classes. That structure is the bare minimum required to perform an optimisation on a leg-cabin.

As for now, that method is called only by RMOL. Indeed, the revenue management component (RMOL) needs very basic set up in order to perform optimisation at leg-level. Hence, there are:

- a dedicated inventory ('XX'),
- the corresponding flight-date (#9999, departing 01/01/1900),

- a leg-date (departing and arriving from/to 'XXX' airport),
-
- a leg-cabin ('X').
-

Most of the data is dummy because RMOL uses only the cabin capacity from that part of the BOM tree.

Parameters:

const CabinCapacity_T& Cabin capacity for revenue management optimisation.

Definition at line 187 of file STDAIR_Service.cpp.

32.154.3.3 void stdair::STDAIR_Service::buildDummyLegSegmentAccesses (BomRoot & *iBomRoot*)

Build the direct accesses between the dummy segment cabins and the dummy leg cabins within the dummy flight dates (the dummy fare family flight date and the classic dummy flight date).

As for now (May 2012), that method is called only by RMOL. It is a substitute for the code doing it automatically located in AirInv. See the AIRINV::InventoryManager::createDirectAccesses command.

Parameters:

BomRoot& Top of the BOM tree, to which the sample should be attached.

Definition at line 204 of file STDAIR_Service.cpp.

32.154.3.4 void stdair::STDAIR_Service::buildSampleTravelSolutionForPricing (TravelSolutionList_T & *ioTravelSolutionList*)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10

Parameters:

TravelSolutionList_T& Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.

Definition at line 215 of file STDAIR_Service.cpp.

32.154.3.5 void stdair::STDAIR_Service::buildSampleTravelSolutions (TravelSolutionList_T & *ioTravelSolutionList*)

Build a sample list of travel solutions.

As of now (March 2011), that list is made of the following travel solutions:

- BA9
- LHR-SYD
- 2011-06-10
- Q
- WTP: 900
- Change fee: 20; Non refundable; Saturday night stay

Parameters:

TravelSolutionList_T& Sample list of travel solution structures. It should be given empty. It is altered with the returned sample.

Definition at line 222 of file STDAIR_Service.cpp.

32.154.3.6 BookingRequestStruct stdair::STDAIR_Service::buildSampleBookingRequest (const bool *isForCRS* = `false`)

Build a sample booking request structure.

As of now (March 2011), the sample booking request is made of the following parameters:

- Return trip (inbound): LHR-SYD (POS: LHR, Channel: DN),
- Departing 10-JUN-2011 around 8:00, staying 7 days
- Requested on 15-MAY-2011 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

As of now (March 2011), the CRS-related booking request is made of the following parameters:

- Return trip (inbound): SIN-BKK (POS: SIN, Channel: IN),
- Departing 30-JAN-2010 around 10:00, staying 7 days
- Requested on 22-JAN-2010 at 10:00
- Economy cabin, 3 persons, FF member
- WTP: 1000.0 EUR
- Dis-utility: 100.0 EUR/hour

Parameters:

const bool isForCRS Whether the sample booking request is for CRS.

Returns:

BookingRequestStruct& Sample booking request structure.

Definition at line 229 of file STDAIR_Service.cpp.

32.154.3.7 void stdair::STDAIR_Service::clonePersistentBom ()

Clone the persistent Bom.

Definition at line 635 of file STDAIR_Service.cpp.

References stdair::FacSupervisor::cleanCloneBomLayer(), and stdair::FacSupervisor::instance().

32.154.3.8 std::string stdair::STDAIR_Service::jsonExportFlightDateList (const AirlineCode_T & *iAirlineCode* = `"all"`, const FlightNumber_T & *iFlightNumber* = `0`) const

Recursively dump, in the returned string and in JSON format, the flight-date list corresponding to the parameters given as input.

Parameters:

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Definition at line 242 of file STDAIR_Service.cpp.

32.154.3.9 std::string stdair::STDAIR_Service::jsonExportFlightDateObjects (const AirlineCode_T & *iAirlineCode*, const FlightNumber_T & *iFlightNumber*, const Date_T & *iDepartureDate*) const

Recursively dump, in the returned string and in JSON format, the detailed flight-date (leg, segments, cabins, classes, ...) corresponding to the parameters given as input.

Parameters:

const AirlineCode_T& Airline code of the flight to dump.

const FlightNumber_T& Flight number of the flight to dump.

const Date_T& Departure date of the flight to dump.

Returns:

std::string Output string in which the BOM tree is JSON-ified.

Definition at line 262 of file STDAIR_Service.cpp.

References stdair::BomRetriever::retrieveFlightDateFromKeySet().

32.154.3.10 std::string stdair::STDAIR_Service::jsonExportEventObject (const EventStruct & *iEventStruct*) const

Recursively dump, in the returned string and in JSON format, the event object.

Returns:

std::string Output string in which the event is JSON-ified.

Definition at line 312 of file STDAIR_Service.cpp.

References stdair::EventType::BKG_REQ, stdair::EventType::BRK_PT, stdair::EventType::CX, stdair::EventStruct::getEventType(), stdair::BomJSONExport::jsonExportBookingRequestObject(), stdair::BomJSONExport::jsonExportBreakPointObject(), stdair::EventType::OPT_NOT_4_FD, stdair::EventType::OPT_NOT_4_NET, stdair::EventType::RM, stdair::EventType::SKD_CHG, and stdair::EventType::SNAPSHOT.

32.154.3.11 std::string stdair::STDAIR_Service::jsonExportConfiguration () const

Dump, in the returned string and in JSON format, the configuration.

Returns:

std::string Output string in which the configuration tree is JSON-ified.

Definition at line 359 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::jsonExport().

32.154.3.12 bool stdair::STDAIR_Service::jsonImportConfiguration (const JSONString & *iJSONString*) const

Extract the configuration ptree from the given JSON-formatted string and add it to the configuration holder

Parameters:

const JSONString& JSON-formatted string.

Returns:

bool State whether the extracting has been successful.

Definition at line 342 of file STDAIR_Service.cpp.

References stdair::BomJSONImport::jsonImportConfig().

32.154.3.13 std::string stdair::STDAIR_Service::list (const AirlineCode_T & *iAirlineCode* = `"all"`, const FlightNumber_T & *iFlightNumber* = `0`) const

Display the list of flight-dates (contained within the BOM tree) corresponding to the parameters given as input.

Parameters:

const AirlineCode& Airline for which the flight-dates should be displayed. If set to "all" (the default), all the inventories will be displayed.

const FlightNumber_T& Flight number for which all the departure dates should be displayed. If set to 0 (the default), all the flight numbers will be displayed.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 428 of file STDAIR_Service.cpp.

32.154.3.14 std::string stdair::STDAIR_Service::listAirportPairDateRange () const

Display the list of aiports pairs and date ranges (contained within the BOM tree)

Parameters:

std::ostream& Output stream in which the airport pairs and date ranges are logged/dumped.

Definition at line 446 of file STDAIR_Service.cpp.

32.154.3.15 bool stdair::STDAIR_Service::check (const AirlineCode_T & *iAirlineCode*, const FlightNumber_T & *iFlightNumber*, const Date_T & *iDepartureDate*) const

Check whether the given flight-date is a valid one.

Parameters:

const stdair::AirlineCode_T& Airline code of the flight to check.

const stdair::FlightNumber_T& Flight number of the flight to check.

const stdair::Date_T& Departure date of the flight to check.

Returns:

bool Whether or not the given flight date is valid.

Definition at line 463 of file STDAIR_Service.cpp.

References stdair::BomRetriever::retrieveFlightDateFromKeySet().

32.154.3.16 bool stdair::STDAIR_Service::check (const AirportCode_T & *ioOrigin*, const AirportCode_T & *ioDestination*, const Date_T & *iDepartureDate*) const

Check whether the given couple airportpair-date is a valid one.

Parameters:

const stdair::AirportCode_T& Origin airport of the fare rule to check.

const stdair::AirportCode_T& Destination airport of the fare rule to check.

const stdair::Date_T& Departure date of the fare rule to check.

Returns:

bool Whether or not the given airportpair-date couple is a valid one.

Definition at line 485 of file STDAIR_Service.cpp.

References stdair::BomRetriever::retrieveDatePeriodListFromKeySet().

32.154.3.17 std::string stdair::STDAIR_Service::configDisplay () const

Display (dump in the returned string) the configuration.

Returns:

std::string Output string in which the configuration is logged/dumped.

Definition at line 508 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::describe().

32.154.3.18 std::string stdair::STDAIR_Service::csvDisplay () const

Recursively display (dump in the returned string) the objects of the persistent BOM tree.

Returns:

std::string Output string in which the persistent BOM tree is logged/dumped.

Definition at line 525 of file STDAIR_Service.cpp.

Referenced by csvDisplay().

32.154.3.19 std::string stdair::STDAIR_Service::csvDisplay (const BomRoot & *iBomRoot*) const

Recursively display (dump in the returned string) the objects of the BOM tree.

Parameters:

const BomRoot& Reference on the BomRoot to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 541 of file STDAIR_Service.cpp.

References csvDisplay().

32.154.3.20 std::string stdair::STDAIR_Service::csvDisplay (const AirlineCode_T & *iAirlineCode*, const FlightNumber_T & *iFlightNumber*, const Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the flight-date corresponding to the parameters given as input.

Parameters:

const AirlineCode_T& Airline code of the flight to display.

const FlightNumber_T& Flight number of the flight to display.

const Date_T& Departure date of the flight to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 555 of file STDAIR_Service.cpp.

References csvDisplay(), and stdair::BomRetriever::retrieveFlightDateFromKeySet().

32.154.3.21 std::string stdair::STDAIR_Service::csvDisplay (const TravelSolutionList_T & *iTravelSolutionList*) const

Display (dump in the returned string) the full list of travel solution structures.

Returns:

std::string Output string in which the list of travel solutions is logged/dumped.

Definition at line 587 of file STDAIR_Service.cpp.

References csvDisplay().

32.154.3.22 std::string stdair::STDAIR_Service::csvDisplay (const AirportCode_T & *iOrigin*, const AirportCode_T & *iDestination*, const Date_T & *iDepartureDate*) const

Recursively display (dump in the returned string) the fare-rules corresponding to the parameters given as input.

Parameters:

const AirportCode_T& Origin airport of the fare-rules to display

const AirportCode_T& Destination airport of the fare-rules to display.

const Date_T& Departure date of the fare-rules to display.

Returns:

std::string Output string in which the BOM tree is logged/dumped.

Definition at line 598 of file STDAIR_Service.cpp.

References csvDisplay(), and stdair::BomRetriever::retrieveDatePeriodListFromKeySet().

32.154.3.23 BomRoot & stdair::STDAIR_Service::getBomRoot () const

Get a reference on the BomRoot object. If the service context has not been initialised, that method throws an exception (failing assertion).

Returns:

BomRoot& Reference on the BomRoot.

Definition at line 128 of file STDAIR_Service.cpp.

32.154.3.24 BomRoot & stdair::STDAIR_Service::getPersistentBomRoot () const

Get a reference on the BomRoot object. If the service context has not been initialised, that method throws an exception (failing assertion).

Returns:

BomRoot& Reference on the BomRoot.

Definition at line 138 of file STDAIR_Service.cpp.

32.154.3.25 BasLogParams stdair::STDAIR_Service::getLogParams () const

Get the log parameters.

Returns:

BasLogParams Copy of the structure holding the log parameters.

Definition at line 148 of file STDAIR_Service.cpp.

32.154.3.26 const BasDBParams & stdair::STDAIR_Service::getDBParams () const

Get the database parameters.

Returns:

const BasDBParams& Reference on the structure holding the database parameters.

Definition at line 153 of file STDAIR_Service.cpp.

32.154.3.27 const ServiceInitialisationType & stdair::STDAIR_Service::getServiceInitialisationType () const

Get the type of initialisation (e.g., not yet, file parsing, sample BOM) which the component (owner of the current STDAIR_Service instance) has gone through.

Returns:

const ServiceInitialisationType& Reference on the type of initialisation (enumeration structure).

Definition at line 163 of file STDAIR_Service.cpp.

32.154.3.28 void stdair::STDAIR_Service::importINIConfig (const ConfigINIFile & *iConfigINIFile*)

Import the configuration INI input file (format cfg).

Parameters:

const ConfigINIFile& INI input file.

Definition at line 375 of file STDAIR_Service.cpp.

32.154.3.29 void stdair::STDAIR_Service::importConfigValue (const std::string & *iValue*, const std::string & *iPath*)

Create the given specified path in the configuration tree and add the corresponding given value (or replace the value if the path already exists).

Parameters:

const std::string& Value to add in the configuration tree.

const std::string& Path to create (or to look for).

Definition at line 391 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::addValue().

32.154.3.30 template<typename ValueType > bool stdair::STDAIR_Service::exportConfigValue (ValueType & *ioValue*, const std::string & *iPath*) `[inline]`

Look for the specified path in the configuration tree and, if existing, try to extract the corresponding value. The type of the value to extract is a template parameter.

Parameters:

ValueType& Value to add in the configuration tree.

const std::string& Path to look for.

Definition at line 552 of file STDAIR_Service.hpp.

References stdair::ConfigHolderStruct::exportValue().

32.154.3.31 void stdair::STDAIR_Service::updateAirlineFeatures ()

Update the airline features objects thanks to the configuration holder.

Definition at line 408 of file STDAIR_Service.cpp.

References stdair::ConfigHolderStruct::updateAirlineFeatures().

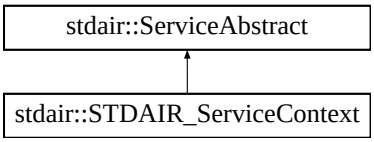
The documentation for this class was generated from the following files:

- stdair/STDAIR_Service.hpp
- stdair/service/STDAIR_Service.cpp

32.155 stdair::STDAIR_ServiceContext Class Reference

Class holding the context of the Stdair services.

`#include <stdair/service/STDAIR_ServiceContext.hpp>`Inheritance diagram for stdair::STDAIR_ServiceContext::



Public Member Functions

- virtual void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Friends

- class STDAIR_Service
- class FacSTDAIRServiceContext

32.155.1 Detailed Description

Class holding the context of the Stdair services.

Definition at line 25 of file STDAIR_ServiceContext.hpp.

32.155.2 Member Function Documentation

32.155.2.1 virtual void stdair::ServiceAbstract::toStream (std::ostream & *ioOut*) const `[inline, virtual, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Definition at line 28 of file ServiceAbstract.hpp.

32.155.2.2 virtual void stdair::ServiceAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Definition at line 35 of file ServiceAbstract.hpp.

Referenced by operator>>().

32.155.3 Friends And Related Function Documentation

32.155.3.1 friend class STDAIR_Service [friend]

The STDAIR_Service class should be the sole class to get access to ServiceContext content: general users do not want to bother with a context interface.

Definition at line 29 of file STDAIR_ServiceContext.hpp.

32.155.3.2 friend class FacSTDAIRServiceContext [friend]

Definition at line 30 of file STDAIR_ServiceContext.hpp.

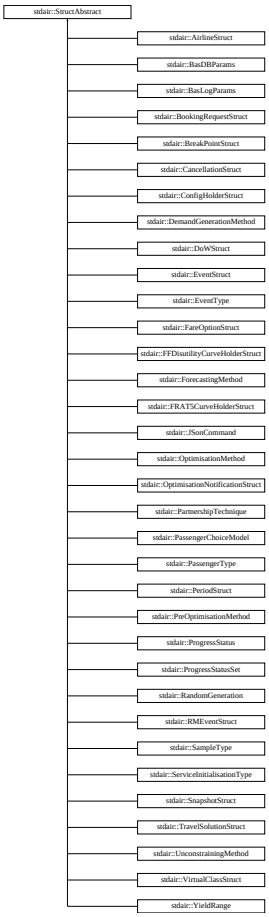
The documentation for this class was generated from the following file:

- stdair/service/STDAIR_ServiceContext.hpp

32.156 stdair::StructAbstract Struct Reference

Base class for the light structures.

`#include <stdair/basic/StructAbstract.hpp>`Inheritance diagram for stdair::StructAbstract::



Public Member Functions

- virtual ~StructAbstract ()
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)
- virtual const std::string describe () const =0

Protected Member Functions

- StructAbstract ()

32.156.1 Detailed Description

Base class for the light structures.

Definition at line 16 of file StructAbstract.hpp.

32.156.2 Constructor & Destructor Documentation

32.156.2.1 virtual stdair::StructAbstract::~StructAbstract () [inline, virtual]

Destructor.

Definition at line 22 of file StructAbstract.hpp.

32.156.2.2 stdair::StructAbstract::StructAbstract () [inline, protected]

Protected Default Constructor to ensure this class is abtract.

Definition at line 49 of file StructAbstract.hpp.

32.156.3 Member Function Documentation

32.156.3.1 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const [inline]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References describe().

32.156.3.2 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

Referenced by operator>>().

32.156.3.3 virtual const std::string stdair::StructAbstract::describe () const [pure virtual]

Display of the structure.

Implemented in stdair::BasDBParams, stdair::BasLogParams, stdair::DemandGenerationMethod, stdair::EventType, stdair::ForecastingMethod, stdair::JSonCommand, stdair::OptimisationMethod, stdair::PartnershipTechnique, stdair::PassengerChoiceModel, stdair::PassengerType, stdair::PreOptimisationMethod, stdair::ProgressStatus, stdair::ProgressStatusSet, stdair::RandomGeneration, stdair::SampleType, stdair::ServiceInitialisationType, stdair::UnconstrainingMethod, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::DoWStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::PeriodStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Referenced by toStream().

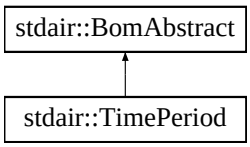
The documentation for this struct was generated from the following file:

- stdair/basic/StructAbstract.hpp

32.157 stdair::TimePeriod Class Reference

Class representing the actual attributes for a fare time-period.

`#include <stdair/bom/TimePeriod.hpp>`Inheritance diagram for stdair::TimePeriod::



Public Types

- typedef TimePeriodKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const Time_T & getTimeRangeStart () const
- const Time_T & getTimeRangeEnd () const
- bool isDepartureTimeValid (const Time_T &) const

Protected Member Functions

- TimePeriod (const Key_T &)
- virtual ~TimePeriod ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.157.1 Detailed Description

Class representing the actual attributes for a fare time-period.

Definition at line 18 of file TimePeriod.hpp.

32.157.2 Member Typedef Documentation

32.157.2.1 typedef TimePeriodKey stdair::TimePeriod::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 28 of file TimePeriod.hpp.

32.157.3 Constructor & Destructor Documentation

32.157.3.1 stdair::TimePeriod::TimePeriod (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 27 of file TimePeriod.cpp.

32.157.3.2 stdair::TimePeriod::~TimePeriod () `[protected, virtual]`

Destructor.

Definition at line 32 of file TimePeriod.cpp.

32.157.4 Member Function Documentation

32.157.4.1 void stdair::TimePeriod::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file TimePeriod.hpp.

References toString().

32.157.4.2 void stdair::TimePeriod::fromStream (std::istream & *ioIn*) `[inline, virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file TimePeriod.hpp.

32.157.4.3 std::string stdair::TimePeriod::toString () const `[virtual]`

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 36 of file TimePeriod.cpp.

References describeKey().

Referenced by toStream().

32.157.4.4 const std::string stdair::TimePeriod::describeKey () const `[inline]`

Get a string describing the key.

Definition at line 58 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::toString().

Referenced by toString().

32.157.4.5 const Key_T& stdair::TimePeriod::getKey () const `[inline]`

Get the primary key (time range start, time range end).

Definition at line 67 of file TimePeriod.hpp.

References _key.

32.157.4.6 BomAbstract* const stdair::TimePeriod::getParent () const `[inline]`

Get a reference on the parent object instance.

Definition at line 74 of file TimePeriod.hpp.

References _parent.

32.157.4.7 const HolderMap_T& stdair::TimePeriod::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file TimePeriod.hpp.

References _holderMap.

32.157.4.8 const Time_T& stdair::TimePeriod::getTimeRangeStart () const [inline]

Get the time range start.

Definition at line 88 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::getTimeRangeStart().

Referenced by isDepartureTimeValid().

32.157.4.9 const Time_T& stdair::TimePeriod::getTimeRangeEnd () const [inline]

Get the time range end

Definition at line 95 of file TimePeriod.hpp.

References _key, and stdair::TimePeriodKey::getTimeRangeEnd().

Referenced by isDepartureTimeValid().

32.157.4.10 bool stdair::TimePeriod::isDepartureTimeValid (const Time_T & *iFlightTime*) const

Check if the given departure time is included in the departure period of the segment path.

Definition at line 44 of file TimePeriod.cpp.

References getTimeRangeEnd(), getTimeRangeStart(), and STDAIR_LOG_DEBUG.

32.157.5 Friends And Related Function Documentation

32.157.5.1 friend class FacBom [friend]

Definition at line 19 of file TimePeriod.hpp.

32.157.5.2 friend class FacCloneBom [friend]

Definition at line 20 of file TimePeriod.hpp.

32.157.5.3 friend class FacBomManager [friend]

Definition at line 21 of file TimePeriod.hpp.

32.157.6 Member Data Documentation

32.157.6.1 Key_T stdair::TimePeriod::_key [protected]

Primary key (flight number and departure date).

Definition at line 133 of file TimePeriod.hpp.

Referenced by describeKey(), getKey(), getTimeRangeEnd(), and getTimeRangeStart().

32.157.6.2 BomAbstract* stdair::TimePeriod::_parent [protected]

Pointer on the parent class (Inventory).

Definition at line 138 of file TimePeriod.hpp.

Referenced by getParent().

32.157.6.3 HolderMap_T stdair::TimePeriod::_holderMap [protected]

Map holding the children.

Definition at line 143 of file TimePeriod.hpp.

Referenced by getHolderMap().

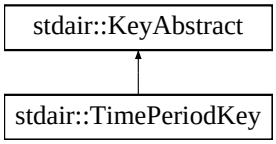
The documentation for this class was generated from the following files:

- stdair/bom/TimePeriod.hpp
- stdair/bom/TimePeriod.cpp

32.158 stdair::TimePeriodKey Struct Reference

Key of time-period.

`#include <stdair/bom/TimePeriodKey.hpp>`Inheritance diagram for stdair::TimePeriodKey::



Public Member Functions

- TimePeriodKey (const Time_T &, const Time_T &)
- TimePeriodKey (const TimePeriodKey &)
- ~TimePeriodKey ()
- const Time_T & getTimeRangeStart () const
- const Time_T & getTimeRangeEnd () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.158.1 Detailed Description

Key of time-period.

Definition at line 15 of file TimePeriodKey.hpp.

32.158.2 Constructor & Destructor Documentation

32.158.2.1 stdair::TimePeriodKey::TimePeriodKey (const Time_T & *iTimeRangeStart*, const Time_T & *iTimeRangeEnd*)

Main constructor.

Definition at line 21 of file TimePeriodKey.cpp.

32.158.2.2 stdair::TimePeriodKey::TimePeriodKey (const TimePeriodKey & *iKey*)

Copy constructor.

Definition at line 28 of file TimePeriodKey.cpp.

32.158.2.3 stdair::TimePeriodKey::~TimePeriodKey ()

Destructor.

Definition at line 34 of file TimePeriodKey.cpp.

32.158.3 Member Function Documentation

32.158.3.1 const Time_T& stdair::TimePeriodKey::getTimeRangeStart () const `[inline]`

Get the time period start.

Definition at line 35 of file TimePeriodKey.hpp.

Referenced by stdair::TimePeriod::getTimeRangeStart().

32.158.3.2 const Time_T& stdair::TimePeriodKey::getTimeRangeEnd () const `[inline]`

Get the time period end.

Definition at line 42 of file TimePeriodKey.hpp.

Referenced by stdair::TimePeriod::getTimeRangeEnd().

32.158.3.3 void stdair::TimePeriodKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 38 of file TimePeriodKey.cpp.

References toString().

32.158.3.4 void stdair::TimePeriodKey::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 43 of file TimePeriodKey.cpp.

32.158.3.5 const std::string stdair::TimePeriodKey::toString () const `[virtual]`

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 47 of file TimePeriodKey.cpp.

Referenced by stdair::TimePeriod::describeKey(), and toStream().

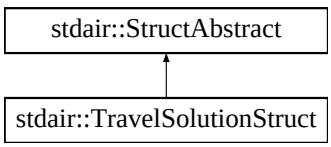
The documentation for this struct was generated from the following files:

- stdair/bom/TimePeriodKey.hpp
- stdair/bom/TimePeriodKey.cpp

32.159 stdair::TravelSolutionStruct Struct Reference

Structure holding the elements of a travel solution.

`#include <stdair/bom/TravelSolutionStruct.hpp>`Inheritance diagram for stdair::TravelSolutionStruct::



Public Member Functions

- const SegmentPath_T & getSegmentPath () const
- const ClassAvailabilityMapHolder_T & getClassAvailabilityMapHolder () const
- const ClassObjectIDMapHolder_T & getClassObjectIDMapHolder () const
- const ClassYieldMapHolder_T & getClassYieldMapHolder () const
- const BidPriceVectorHolder_T & getBidPriceVectorHolder () const

- const ClassBpvMapHolder_T & getClassBpvMapHolder () const
- const FareOptionList_T & getFareOptionList () const
- FareOptionList_T & getFareOptionListRef ()
- const FareOptionStruct & getChosenFareOption () const
- void addSegment (const std::string &)
- void addClassAvailabilityMap (const ClassAvailabilityMap_T &)
- void addClassObjectIDMap (const ClassObjectIDMap_T &)
- void addClassYieldMap (const ClassYieldMap_T &)
- void addBidPriceVector (const BidPriceVector_T &)
- void addClassBpvMap (const ClassBpvMap_T &)
- void addFareOption (const FareOptionStruct &)
- void setChosenFareOption (const FareOptionStruct &iChosenFO)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- const std::string display () const
- const std::string describeSegmentPath () const
- TravelSolutionStruct ()
- ~TravelSolutionStruct ()

32.159.1 Detailed Description

Structure holding the elements of a travel solution.

Definition at line 24 of file TravelSolutionStruct.hpp.

32.159.2 Constructor & Destructor Documentation

32.159.2.1 stdair::TravelSolutionStruct::TravelSolutionStruct ()

Default constructor.

Definition at line 15 of file TravelSolutionStruct.cpp.

32.159.2.2 stdair::TravelSolutionStruct::~TravelSolutionStruct ()

Destructor.

Definition at line 19 of file TravelSolutionStruct.cpp.

32.159.3 Member Function Documentation

32.159.3.1 const SegmentPath_T& stdair::TravelSolutionStruct::getSegmentPath () const [inline]

Get the segment path.

Definition at line 28 of file TravelSolutionStruct.hpp.

32.159.3.2 const ClassAvailabilityMapHolder_T& stdair::TravelSolutionStruct::getClassAvailabilityMapHolder () const [inline]

Get the holder of avalabilities.

Definition at line 33 of file TravelSolutionStruct.hpp.

32.159.3.3 const ClassObjectIDMapHolder_T& stdair::TravelSolutionStruct::getClassObjectIDMapHolder () const [inline]

Get the holder of object ID's.

Definition at line 38 of file TravelSolutionStruct.hpp.

32.159.3.4 const ClassYieldMapHolder_T& stdair::TravelSolutionStruct::getClassYieldMapHolder () const [inline]

Get the holder of yields.

Definition at line 43 of file TravelSolutionStruct.hpp.

32.159.3.5 const BidPriceVectorHolder_T& stdair::TravelSolutionStruct::getBidPriceVectorHolder () const [inline]

Get the holder of bid price vectors.

Definition at line 48 of file TravelSolutionStruct.hpp.

32.159.3.6 const ClassBpvMapHolder_T& stdair::TravelSolutionStruct::getClassBpvMapHolder () const [inline]

Get the holder of class - bid price reference.

Definition at line 53 of file TravelSolutionStruct.hpp.

32.159.3.7 const FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionList () const [inline]

Get the list of fare options.

Definition at line 58 of file TravelSolutionStruct.hpp.

32.159.3.8 FareOptionList_T& stdair::TravelSolutionStruct::getFareOptionListRef () [inline]

Get the non-const list of fare options.

Definition at line 63 of file TravelSolutionStruct.hpp.

32.159.3.9 const FareOptionStruct& stdair::TravelSolutionStruct::getChosenFareOption () const [inline]

Get the chosen fare option.

Definition at line 68 of file TravelSolutionStruct.hpp.

32.159.3.10 void stdair::TravelSolutionStruct::addSegment (const std::string & *iKey*)

Add a segment key to the segment path.

Definition at line 154 of file TravelSolutionStruct.cpp.

32.159.3.11 void stdair::TravelSolutionStruct::addClassAvailabilityMap (const ClassAvailabilityMap_T & *iMap*)

Add a class availability map.

Definition at line 160 of file TravelSolutionStruct.cpp.

32.159.3.12 void stdair::TravelSolutionStruct::addClassObjectIDMap (const ClassObjectIDMap_T & *iMap*)

Add a class object ID map.

Definition at line 166 of file TravelSolutionStruct.cpp.

32.159.3.13 void stdair::TravelSolutionStruct::addClassYieldMap (const ClassYieldMap_T & *iMap*)

Add a class yield map.

Definition at line 172 of file TravelSolutionStruct.cpp.

32.159.3.14 void stdair::TravelSolutionStruct::addBidPriceVector (const BidPriceVector_T & *iBpv*)

Add a bid price vector.

Definition at line 178 of file TravelSolutionStruct.cpp.

32.159.3.15 void stdair::TravelSolutionStruct::addClassBpvMap (const ClassBpvMap_T & *iMap*)

Add a class bpv reference map.

Definition at line 184 of file TravelSolutionStruct.cpp.

32.159.3.16 void stdair::TravelSolutionStruct::addFareOption (const FareOptionStruct & *iFareOption*)

Add a fare option.

Definition at line 190 of file TravelSolutionStruct.cpp.

32.159.3.17 void stdair::TravelSolutionStruct::setChosenFareOption (const FareOptionStruct & *iChosenFO*) [inline]

Set the chosen fare option.

Definition at line 97 of file TravelSolutionStruct.hpp.

32.159.3.18 void stdair::TravelSolutionStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 23 of file TravelSolutionStruct.cpp.

References describe().

32.159.3.19 void stdair::TravelSolutionStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 28 of file TravelSolutionStruct.cpp.

32.159.3.20 const std::string stdair::TravelSolutionStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 53 of file TravelSolutionStruct.cpp.

References stdair::FareOptionStruct::describe(), stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

Referenced by toStream().

32.159.3.21 const std::string stdair::TravelSolutionStruct::display () const

Display of the structure.

Definition at line 95 of file TravelSolutionStruct.cpp.

References stdair::FareOptionStruct::display(), stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

32.159.3.22 const std::string stdair::TravelSolutionStruct::describeSegmentPath () const

Display only the segment path.

Definition at line 32 of file TravelSolutionStruct.cpp.

References stdair::BomKeyManager::extractKeys(), and stdair::ParsedKey::toString().

The documentation for this struct was generated from the following files:

- stdair/bom/TravelSolutionStruct.hpp
- stdair/bom/TravelSolutionStruct.cpp

32.160 soci::type_conversion< stdair::AirlineStruct > Struct Template Reference

```
#include <stdair/dbadaptor/DbaAirline.hpp>
```

Public Types

- typedef values base_type

Static Public Member Functions

- static void from_base (values const &iAirlineValues, indicator, stdair::AirlineStruct &ioAirline)
- static void to_base (const stdair::AirlineStruct &iAirline, values &ioAirlineValues, indicator &ioIndicator)

32.160.1 Detailed Description

template<> struct soci::type_conversion< stdair::AirlineStruct >

Specify how the AirlineStruct struct can be converted to (resp. from) values stored into (resp. retrieved from) database, using the SOCI framework.

Definition at line 25 of file DbaAirline.hpp.

32.160.2 Member Typedef Documentation

32.160.2.1 typedef values soci::type_conversion< stdair::AirlineStruct >::base_type

Definition at line 27 of file DbaAirline.hpp.

32.160.3 Member Function Documentation

32.160.3.1 void soci::type_conversion< stdair::AirlineStruct >::from_base (values const & *iAirlineValues*, indicator, stdair::AirlineStruct & *ioAirline*) [static]

Fill an Airline object from the database values.

Definition at line 17 of file DbaAirline.cpp.

References stdair::AirlineStruct::setAirlineCode(), and stdair::AirlineStruct::setAirlineName().

32.160.3.2 void soci::type_conversion< stdair::AirlineStruct >::to_base (const stdair::AirlineStruct & *iAirline*, values & *ioAirlineValues*, indicator & *ioIndicator*) [static]

Fill the database values from an Airline object.

Definition at line 30 of file DbaAirline.cpp.

References stdair::AirlineStruct::getAirlineCode(), and stdair::AirlineStruct::getAirlineName().

The documentation for this struct was generated from the following files:

- stdair/dbadaptor/DbaAirline.hpp
- stdair/dbadaptor/DbaAirline.cpp

32.161 TypeWithSize< size > Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef void UInt

32.161.1 Detailed Description

template<size_t size> class TypeWithSize< size >

Definition at line 54 of file float_utils_google.hpp.

32.161.2 Member Typedef Documentation

32.161.2.1 template<size_t size> typedef void TypeWithSize< size >::UInt

Definition at line 58 of file float_utils_google.hpp.

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

- stdair/basic/float_utils_google.hpp

32.162 TypeWithSize< 4 > Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef int Int
- typedef unsigned int UInt

32.162.1 Detailed Description

template<> class TypeWithSize< 4 >

Definition at line 63 of file float_utils_google.hpp.

32.162.2 Member Typedef Documentation

32.162.2.1 typedef int TypeWithSize< 4 >::Int

Definition at line 69 of file float_utils_google.hpp.

32.162.2.2 typedef unsigned int TypeWithSize< 4 >::UInt

Definition at line 70 of file float_utils_google.hpp.

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

- stdair/basic/float_utils_google.hpp

32.163 TypeWithSize< 8 > Class Template Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Public Types

- typedef long long Int
- typedef unsigned long long UInt

32.163.1 Detailed Description

template<> class TypeWithSize< 8 >

Definition at line 75 of file float_utils_google.hpp.

32.163.2 Member Typedef Documentation

32.163.2.1 typedef long long TypeWithSize< 8 >::Int

Definition at line 81 of file float_utils_google.hpp.

32.163.2.2 typedef unsigned long long TypeWithSize< 8 >::UInt

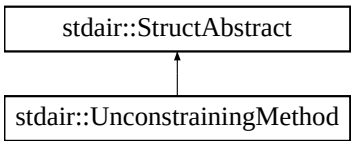
Definition at line 82 of file float_utils_google.hpp.

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

- stdair/basic/float_utils_google.hpp

32.164 stdair::UnconstrainingMethod Struct Reference

`#include <stdair/basic/UnconstrainingMethod.hpp>`Inheritance diagram for stdair::UnconstrainingMethod::



Public Types

- enum EN_UnconstrainingMethod { EM = 0, LAST_VALUE }

Public Member Functions

- EN_UnconstrainingMethod getMethod () const
- std::string getMethodAsString () const
- const std::string describe () const
- bool operator== (const EN_UnconstrainingMethod &) const
- UnconstrainingMethod (const EN_UnconstrainingMethod &)
- UnconstrainingMethod (const char iMethod)
- UnconstrainingMethod (const UnconstrainingMethod &)
- void toStream (std::ostream &ioOut) const
- virtual void fromStream (std::istream &ioIn)

Static Public Member Functions

- static const std::string & getLabel (const EN_UnconstrainingMethod &)
- static char getMethodLabel (const EN_UnconstrainingMethod &)
- static std::string getMethodLabelAsString (const EN_UnconstrainingMethod &)
- static std::string describeLabels ()

32.164.1 Detailed Description

Enumeration of unconstraining methods.

Definition at line 15 of file UnconstrainingMethod.hpp.

32.164.2 Member Enumeration Documentation

32.164.2.1 enum stdair::UnconstrainingMethod::EN_UnconstrainingMethod

Enumerator:

EM

LAST_VALUE

Definition at line 17 of file UnconstrainingMethod.hpp.

32.164.3 Constructor & Destructor Documentation

32.164.3.1 stdair::UnconstrainingMethod::UnconstrainingMethod (const EN_UnconstrainingMethod & *iUnconstrainingMethod*)

Constructor.

Definition at line 36 of file UnconstrainingMethod.cpp.

32.164.3.2 stdair::UnconstrainingMethod::UnconstrainingMethod (const char *iMethod*)

Constructor.

Definition at line 41 of file UnconstrainingMethod.cpp.

References describeLabels(), EM, and LAST_VALUE.

32.164.3.3 stdair::UnconstrainingMethod::UnconstrainingMethod (const UnconstrainingMethod & *iUnconstrainingMethod*)

Default copy constructor.

Definition at line 30 of file UnconstrainingMethod.cpp.

32.164.4 Member Function Documentation

32.164.4.1 const std::string & stdair::UnconstrainingMethod::getLabel (const EN_UnconstrainingMethod & *iMethod*) `[static]`

Get the label as a string (e.g., "Expectation-Maximisation")

Definition at line 58 of file UnconstrainingMethod.cpp.

32.164.4.2 char stdair::UnconstrainingMethod::getMethodLabel (const EN_UnconstrainingMethod & *iMethod*) `[static]`

Get the label as a single char (e.g., 'T' or 'B').

Definition at line 63 of file UnconstrainingMethod.cpp.

32.164.4.3 std::string stdair::UnconstrainingMethod::getMethodLabelAsString (const EN_UnconstrainingMethod & *iMethod*) `[static]`

Get the label as a string of a single char (e.g., "T" or "B").

Definition at line 69 of file UnconstrainingMethod.cpp.

32.164.4.4 std::string stdair::UnconstrainingMethod::describeLabels () `[static]`

List the labels.

Definition at line 76 of file UnconstrainingMethod.cpp.

References LAST_VALUE.

Referenced by UnconstrainingMethod().

32.164.4.5 UnconstrainingMethod::EN_UnconstrainingMethod stdair::UnconstrainingMethod::getMethod () const

Get the enumerated value.

Definition at line 88 of file UnconstrainingMethod.cpp.

Referenced by stdair::AirlineFeature::getUnconstrainingMethod().

32.164.4.6 std::string stdair::UnconstrainingMethod::getMethodAsString () const

Get the enumerated value as a short string (e.g., "T" or "B").

Definition at line 93 of file UnconstrainingMethod.cpp.

32.164.4.7 const std::string stdair::UnconstrainingMethod::describe () const `[virtual]`

Give a description of the structure (e.g., "Expectation-Maximisation").

Implements stdair::StructAbstract.

Definition at line 100 of file UnconstrainingMethod.cpp.

32.164.4.8 bool stdair::UnconstrainingMethod::operator== (const EN_UnconstrainingMethod & *iMethod*) const

Comparison operator.

Definition at line 108 of file UnconstrainingMethod.cpp.

32.164.4.9 void stdair::StructAbstract::toStream (std::ostream & *ioOut*) const `[inline, inherited]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented in stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 29 of file StructAbstract.hpp.

References stdair::StructAbstract::describe().

32.164.4.10 virtual void stdair::StructAbstract::fromStream (std::istream & *ioIn*) `[inline, virtual, inherited]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented in stdair::ProgressStatusSet, stdair::YieldRange, stdair::AirlineStruct, stdair::BookingRequestStruct, stdair::BreakPointStruct, stdair::CancellationStruct, stdair::ConfigHolderStruct, stdair::EventStruct, stdair::FareOptionStruct, stdair::FFDisutilityCurveHolderStruct, stdair::FRAT5CurveHolderStruct, stdair::OptimisationNotificationStruct, stdair::RMEventStruct, stdair::SnapshotStruct, stdair::TravelSolutionStruct, and stdair::VirtualClassStruct.

Definition at line 38 of file StructAbstract.hpp.

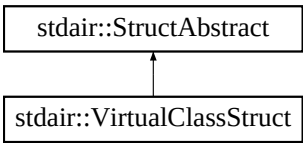
Referenced by operator>>().

The documentation for this struct was generated from the following files:

- stdair/basic/UnconstrainingMethod.hpp
- stdair/basic/UnconstrainingMethod.cpp

32.165 stdair::VirtualClassStruct Struct Reference

`#include <stdair/bom/VirtualClassStruct.hpp>`Inheritance diagram for stdair::VirtualClassStruct::



Public Member Functions

- const BookingClassList_T & getBookingClassList () const
- const Yield_T & getYield () const
- const MeanValue_T & getMean () const
- const StdDevValue_T & getStdDev () const
- const BookingLimit_T & getCumulatedBookingLimit () const

- const ProtectionLevel_T & getCumulatedProtection () const
- const GeneratedDemandVector_T getGeneratedDemandVector () const
- void setYield (const Yield_T &iYield)
- void setMean (const MeanValue_T &iMean)
- void setStdDev (const StdDevValue_T &iStdDev)
- void setCumulatedBookingLimit (const BookingLimit_T &iBL)
- void setCumulatedProtection (const ProtectionLevel_T &iP)
- void addBookingClass (BookingClass &iBookingClass)
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string describe () const
- VirtualClassStruct (const VirtualClassStruct &)
- VirtualClassStruct (const BookingClassList_T &)
- ~VirtualClassStruct ()

32.165.1 Detailed Description

Structure holding the elements of a virtual class.

Definition at line 24 of file VirtualClassStruct.hpp.

32.165.2 Constructor & Destructor Documentation

32.165.2.1 stdair::VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct & *iVC*)

Default copy constructor.

Definition at line 19 of file VirtualClassStruct.cpp.

32.165.2.2 stdair::VirtualClassStruct::VirtualClassStruct (const BookingClassList_T & *ioBookingClassList*)

Constructor.

Definition at line 26 of file VirtualClassStruct.cpp.

32.165.2.3 stdair::VirtualClassStruct::~VirtualClassStruct ()

Destructor.

Definition at line 31 of file VirtualClassStruct.cpp.

32.165.3 Member Function Documentation

32.165.3.1 const BookingClassList_T& stdair::VirtualClassStruct::getBookingClassList () const `[inline]`

Get the list of booking class.

Definition at line 28 of file VirtualClassStruct.hpp.

32.165.3.2 const Yield_T& stdair::VirtualClassStruct::getYield () const [inline]

Get the yield (average price paid for that virtual class).

Definition at line 33 of file VirtualClassStruct.hpp.

Referenced by stdair::LegCabin::displayVirtualClassList().

32.165.3.3 const MeanValue_T& stdair::VirtualClassStruct::getMean () const [inline]

Get the mean value of the demand distribution.

Definition at line 38 of file VirtualClassStruct.hpp.

32.165.3.4 const StdDevValue_T& stdair::VirtualClassStruct::getStdDev () const [inline]

Get the standard deviation of the demand distribution.

Definition at line 43 of file VirtualClassStruct.hpp.

32.165.3.5 const BookingLimit_T& stdair::VirtualClassStruct::getCumulatedBookingLimit () const [inline]

Get the booking limit of the class.

Definition at line 48 of file VirtualClassStruct.hpp.

Referenced by stdair::LegCabin::displayVirtualClassList().

32.165.3.6 const ProtectionLevel_T& stdair::VirtualClassStruct::getCumulatedProtection () const [inline]

Get the protection level of the class.

Definition at line 53 of file VirtualClassStruct.hpp.

Referenced by stdair::LegCabin::displayVirtualClassList().

32.165.3.7 const GeneratedDemandVector_T stdair::VirtualClassStruct::getGeneratedDemandVector () const

Get the generated demand sample vector for Monte-Carlo method.

Definition at line 54 of file VirtualClassStruct.cpp.

References stdair::BookingClass::getGeneratedDemandVector().

32.165.3.8 void stdair::VirtualClassStruct::setYield (const Yield_T & *iYield*) [inline]

Set the yield (average price paid for that virtual class).

Definition at line 63 of file VirtualClassStruct.hpp.

32.165.3.9 void stdair::VirtualClassStruct::setMean (const MeanValue_T & *iMean*) [inline]

Set the mean value of the demand distribution.

Definition at line 68 of file VirtualClassStruct.hpp.

32.165.3.10 void stdair::VirtualClassStruct::setStdDev (const StdDevValue_T & *iStdDev*) [inline]

Set the standard deviation of the demand distribution.

Definition at line 73 of file VirtualClassStruct.hpp.

32.165.3.11 void stdair::VirtualClassStruct::setCumulatedBookingLimit (const BookingLimit_T & *iBL*) [inline]

Set the booking limit of the class.

Definition at line 78 of file VirtualClassStruct.hpp.

32.165.3.12 void stdair::VirtualClassStruct::setCumulatedProtection (const ProtectionLevel_T & *iP*) [inline]

Set the protection level of the class.

Definition at line 83 of file VirtualClassStruct.hpp.

32.165.3.13 void stdair::VirtualClassStruct::addBookingClass (BookingClass & *iBookingClass*) [inline]

Add a booking class to the list of booking classes. Note: it is not a link Parent/Child so we don't use the FacBom. The Virtual Classes are not bom objects because the optimiser needs to build them before each optimisation.

Definition at line 92 of file VirtualClassStruct.hpp.

32.165.3.14 void stdair::VirtualClassStruct::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 36 of file VirtualClassStruct.cpp.

References describe().

32.165.3.15 void stdair::VirtualClassStruct::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 41 of file VirtualClassStruct.cpp.

32.165.3.16 const std::string stdair::VirtualClassStruct::describe () const [virtual]

Display of the structure.

Implements stdair::StructAbstract.

Definition at line 45 of file VirtualClassStruct.cpp.

Referenced by toStream().

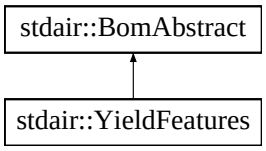
The documentation for this struct was generated from the following files:

- stdair/bom/VirtualClassStruct.hpp
- stdair/bom/VirtualClassStruct.cpp

32.166 stdair::YieldFeatures Class Reference

Class representing the actual attributes for a yield date-period.

`#include <stdair/bom/YieldFeatures.hpp>`Inheritance diagram for stdair::YieldFeatures::



Public Types

- typedef YieldFeaturesKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- BomAbstract *const getParent () const
- const HolderMap_T & getHolderMap () const
- const CabinCode_T & getCabinCode () const
- const TripType_T & getTripType () const
- bool isTripTypeValid (const TripType_T &) const

Protected Member Functions

- YieldFeatures (const Key_T &)
- virtual ~YieldFeatures ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent
- HolderMap_T _holderMap

Friends

- class FacBom
- class FacCloneBom
- class FacBomManager

32.166.1 Detailed Description

Class representing the actual attributes for a yield date-period.

Definition at line 19 of file YieldFeatures.hpp.

32.166.2 Member Typedef Documentation

32.166.2.1 typedef YieldFeaturesKey stdair::YieldFeatures::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 29 of file YieldFeatures.hpp.

32.166.3 Constructor & Destructor Documentation

32.166.3.1 stdair::YieldFeatures::YieldFeatures (const Key_T & *iKey*) `[protected]`

Main constructor.

Definition at line 28 of file YieldFeatures.cpp.

32.166.3.2 stdair::YieldFeatures::~YieldFeatures () `[protected, virtual]`

Destructor.

Definition at line 33 of file YieldFeatures.cpp.

32.166.4 Member Function Documentation

32.166.4.1 void stdair::YieldFeatures::toStream (std::ostream & *ioOut*) const `[inline, virtual]`

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 38 of file YieldFeatures.hpp.

References toString().

32.166.4.2 void stdair::YieldFeatures::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 47 of file YieldFeatures.hpp.

32.166.4.3 std::string stdair::YieldFeatures::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 37 of file YieldFeatures.cpp.

References describeKey().

Referenced by toStream().

32.166.4.4 const std::string stdair::YieldFeatures::describeKey () const [inline]

Get a string describing the key.

Definition at line 58 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::toString().

Referenced by toString().

32.166.4.5 const Key_T& stdair::YieldFeatures::getKey () const [inline]

Get the primary key (trip type, cabin code).

Definition at line 67 of file YieldFeatures.hpp.

References _key.

32.166.4.6 BomAbstract* const stdair::YieldFeatures::getParent () const [inline]

Get a reference on the parent object instance.

Definition at line 74 of file YieldFeatures.hpp.

References _parent.

32.166.4.7 const HolderMap_T& stdair::YieldFeatures::getHolderMap () const [inline]

Get a reference on the children holder.

Definition at line 81 of file YieldFeatures.hpp.

References _holderMap.

32.166.4.8 const CabinCode_T& stdair::YieldFeatures::getCabinCode () const [inline]

Get the cabin code.

Definition at line 88 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::getCabinCode().

32.166.4.9 const TripType_T& stdair::YieldFeatures::getTripType () const [inline]

Get the trip type.

Definition at line 95 of file YieldFeatures.hpp.

References _key, and stdair::YieldFeaturesKey::getTripType().

Referenced by isTripTypeValid().

32.166.4.10 bool stdair::YieldFeatures::isTripTypeValid (const TripType_T & *iBookingRequestTripType*) const

Check whether the fare rule trip type corresponds to the booking request trip type.

Definition at line 45 of file YieldFeatures.cpp.

References getTripType(), stdair::TRIP_TYPE_INBOUND, stdair::TRIP_TYPE_OUTBOUND, and stdair::TRIP_TYPE_ROUND_TRIP.

32.166.5 Friends And Related Function Documentation

32.166.5.1 friend class FacBom [friend]

Definition at line 20 of file YieldFeatures.hpp.

32.166.5.2 friend class FacCloneBom [friend]

Definition at line 21 of file YieldFeatures.hpp.

32.166.5.3 friend class FacBomManager [friend]

Definition at line 22 of file YieldFeatures.hpp.

32.166.6 Member Data Documentation

32.166.6.1 Key_T stdair::YieldFeatures::_key [protected]

Primary key (flight number and departure date).

Definition at line 138 of file YieldFeatures.hpp.

Referenced by describeKey(), getCabinCode(), getKey(), and getTripType().

32.166.6.2 BomAbstract* stdair::YieldFeatures::_parent [protected]

Pointer on the parent class.

Definition at line 143 of file YieldFeatures.hpp.

Referenced by getParent().

32.166.6.3 HolderMap_T stdair::YieldFeatures::_holderMap [protected]

Map holding the children.

Definition at line 148 of file YieldFeatures.hpp.

Referenced by getHolderMap().

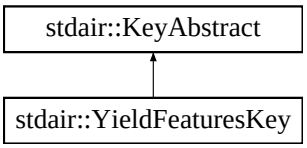
The documentation for this class was generated from the following files:

- stdair/bom/YieldFeatures.hpp
- stdair/bom/YieldFeatures.cpp

32.167 stdair::YieldFeaturesKey Struct Reference

Key of date-period.

`#include <stdair/bom/YieldFeaturesKey.hpp>`Inheritance diagram for stdair::YieldFeaturesKey::



Public Member Functions

- YieldFeaturesKey (const TripType_T &, const CabinCode_T &)
- YieldFeaturesKey (const YieldFeaturesKey &)
- ~YieldFeaturesKey ()
- const TripType_T & getTripType () const
- const CabinCode_T & getCabinCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.167.1 Detailed Description

Key of date-period.

Definition at line 18 of file YieldFeaturesKey.hpp.

32.167.2 Constructor & Destructor Documentation

32.167.2.1 stdair::YieldFeaturesKey::YieldFeaturesKey (const TripType_T & *iTripType*, const CabinCode_T & *iCabin*)

Main constructor.

Definition at line 21 of file YieldFeaturesKey.cpp.

32.167.2.2 stdair::YieldFeaturesKey::YieldFeaturesKey (const YieldFeaturesKey & *iKey*)

Copy constructor.

Definition at line 27 of file YieldFeaturesKey.cpp.

32.167.2.3 stdair::YieldFeaturesKey::~YieldFeaturesKey ()

Destructor.

Definition at line 32 of file YieldFeaturesKey.cpp.

32.167.3 Member Function Documentation

32.167.3.1 const TripType_T& stdair::YieldFeaturesKey::getTripType () const `[inline]`

Get the fare trip type.

Definition at line 44 of file YieldFeaturesKey.hpp.

Referenced by stdair::YieldFeatures::getTripType().

32.167.3.2 const CabinCode_T& stdair::YieldFeaturesKey::getCabinCode () const `[inline]`

Get the cabin.

Definition at line 51 of file YieldFeaturesKey.hpp.

Referenced by stdair::YieldFeatures::getCabinCode().

32.167.3.3 void stdair::YieldFeaturesKey::toStream (std::ostream & *ioOut*) const `[virtual]`

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 36 of file YieldFeaturesKey.cpp.

References toString().

32.167.3.4 void stdair::YieldFeaturesKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 41 of file YieldFeaturesKey.cpp.

32.167.3.5 const std::string stdair::YieldFeaturesKey::toString () const [virtual]

Get the serialised version of the Business Object Key. That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

Reimplemented from stdair::KeyAbstract.

Definition at line 45 of file YieldFeaturesKey.cpp.

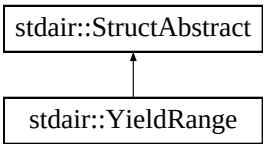
Referenced by stdair::YieldFeatures::describeKey(), and toStream().

The documentation for this struct was generated from the following files:

- stdair/bom/YieldFeaturesKey.hpp
- stdair/bom/YieldFeaturesKey.cpp

32.168 stdair::YieldRange Class Reference

`#include <stdair/basic/YieldRange.hpp>`Inheritance diagram for stdair::YieldRange::



Public Member Functions

- YieldRange ()
- YieldRange (const YieldRange &)
- YieldRange (const Yield_T iUpperYield)
- YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield)
- YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield, const Yield_T iLowerYield)
- virtual ~YieldRange ()
- Yield_T getUpperYield () const

- Yield_T getAverageYield () const
- Yield_T getLowerYield () const
- void setUpperYield (const Yield_T iUpperYield)
- void setAverageYield (const Yield_T iAverageYield)
- void setLowerYield (const Yield_T iLowerYield)
- void toStream (std::ostream &) const
- void fromStream (std::istream &)
- const std::string describe () const

32.168.1 Detailed Description

Class representing a range of yields.

Typically, bookings are priced according to rules (e.g., fare rules), leading to slight variations of revenues for a given product. The "yield range" captures the extent of revenues earned for a given product.

When no average and lower yields are defined, they are assumed to be equal to the upper yield.

Note that the lower yield is generally not defined, as it corresponds to the upper yield of the lower yield range.

Definition at line 23 of file YieldRange.hpp.

32.168.2 Constructor & Destructor Documentation

32.168.2.1 stdair::YieldRange::YieldRange ()

Constructors.

Definition at line 13 of file YieldRange.cpp.

32.168.2.2 stdair::YieldRange::YieldRange (const YieldRange & *iYieldRange*)

Definition at line 20 of file YieldRange.cpp.

32.168.2.3 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*)

Definition at line 27 of file YieldRange.cpp.

32.168.2.4 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*, const Yield_T *iAverageYield*)

Definition at line 33 of file YieldRange.cpp.

32.168.2.5 stdair::YieldRange::YieldRange (const Yield_T *iUpperYield*, const Yield_T *iAverageYield*, const Yield_T *iLowerYield*)

Definition at line 40 of file YieldRange.cpp.

32.168.2.6 stdair::YieldRange::~YieldRange () `[virtual]`

Constructors.

Definition at line 48 of file YieldRange.cpp.

32.168.3 Member Function Documentation

32.168.3.1 Yield_T stdair::YieldRange::getUpperYield () const `[inline]`

Getter for the upper yield of the range.

Definition at line 39 of file YieldRange.hpp.

32.168.3.2 Yield_T stdair::YieldRange::getAverageYield () const `[inline]`

Getter for the average yield of the range.

Definition at line 43 of file YieldRange.hpp.

32.168.3.3 Yield_T stdair::YieldRange::getLowerYield () const `[inline]`

Getter for the lower yield of the range.

Definition at line 47 of file YieldRange.hpp.

32.168.3.4 void stdair::YieldRange::setUpperYield (const Yield_T *iUpperYield*) `[inline]`

Setter for the upper yield of the range.

Definition at line 53 of file YieldRange.hpp.

32.168.3.5 void stdair::YieldRange::setAverageYield (const Yield_T *iAverageYield*) `[inline]`

Setter for the average yield of the range.

Definition at line 57 of file YieldRange.hpp.

32.168.3.6 void stdair::YieldRange::setLowerYield (const Yield_T *iLowerYield*) `[inline]`

Setter for the lower yield of the range.

Definition at line 61 of file YieldRange.hpp.

32.168.3.7 void stdair::YieldRange::toStream (std::ostream & *ioOut*) const

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::StructAbstract.

Definition at line 52 of file YieldRange.cpp.

32.168.3.8 void stdair::YieldRange::fromStream (std::istream & *ioIn*) `[virtual]`

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::StructAbstract.

Definition at line 58 of file YieldRange.cpp.

32.168.3.9 const std::string stdair::YieldRange::describe () const `[virtual]`

Display of the structure.

Implements stdair::StructAbstract.

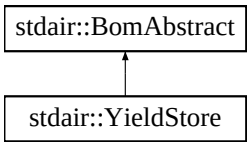
Definition at line 62 of file YieldRange.cpp.

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

- stdair/basic/YieldRange.hpp
- stdair/basic/YieldRange.cpp

32.169 stdair::YieldStore Class Reference

`#include <stdair/bom/YieldStore.hpp>`Inheritance diagram for stdair::YieldStore::



Public Types

- typedef YieldStoreKey Key_T

Public Member Functions

- void toStream (std::ostream &ioOut) const
- BomAbstract *const getParent () const
- void fromStream (std::istream &ioIn)
- std::string toString () const
- const std::string describeKey () const
- const Key_T & getKey () const
- const AirlineCode_T & getAirlineCode () const

Protected Member Functions

- YieldStore (const Key_T &)
- YieldStore (const YieldStore &)
- ~YieldStore ()

Protected Attributes

- Key_T _key
- BomAbstract * _parent

Friends

- class FacBom
- class FacBomManager

32.169.1 Detailed Description

Class representing the actual attributes for an airline YieldStore.

Definition at line 18 of file YieldStore.hpp.

32.169.2 Member Typedef Documentation

32.169.2.1 typedef YieldStoreKey stdair::YieldStore::Key_T

Definition allowing to retrieve the associated BOM key type.

Definition at line 25 of file YieldStore.hpp.

32.169.3 Constructor & Destructor Documentation

32.169.3.1 stdair::YieldStore::YieldStore (const Key_T & *iKey*) `[protected]`

Default constructors.

Definition at line 13 of file YieldStore.cpp.

32.169.3.2 stdair::YieldStore::YieldStore (const YieldStore &) `[protected]`

32.169.3.3 stdair::YieldStore::~YieldStore () [protected]

Destructor.

Definition at line 17 of file YieldStore.cpp.

32.169.4 Member Function Documentation

32.169.4.1 void stdair::YieldStore::toStream (std::ostream & *ioOut*) const [inline, virtual]

Dump a Business Object into an output stream.

Parameters:

ostream& the output stream.

Implements stdair::BomAbstract.

Definition at line 31 of file YieldStore.hpp.

References toString().

32.169.4.2 BomAbstract* const stdair::YieldStore::getParent () const [inline]

Get the parent object.

Definition at line 34 of file YieldStore.hpp.

References _parent.

32.169.4.3 void stdair::YieldStore::fromStream (std::istream & *ioIn*) [inline, virtual]

Read a Business Object from an input stream.

Parameters:

istream& the input stream.

Implements stdair::BomAbstract.

Definition at line 38 of file YieldStore.hpp.

32.169.4.4 std::string stdair::YieldStore::toString () const [virtual]

Get the serialised version of the Business Object.

Implements stdair::BomAbstract.

Definition at line 21 of file YieldStore.cpp.

References _key, and stdair::YieldStoreKey::toString().

Referenced by toStream().

32.169.4.5 const std::string stdair::YieldStore::describeKey () const [inline]

Get a string describing the key.

Definition at line 44 of file YieldStore.hpp.

References _key, and stdair::YieldStoreKey::toString().

32.169.4.6 const Key_T& stdair::YieldStore::getKey () const [inline]

Get the YieldStore key.

Definition at line 49 of file YieldStore.hpp.

References _key.

32.169.4.7 const AirlineCode_T& stdair::YieldStore::getAirlineCode () const [inline]

Get the airline code.

Definition at line 52 of file YieldStore.hpp.

References _key, and stdair::YieldStoreKey::getAirlineCode().

32.169.5 Friends And Related Function Documentation

32.169.5.1 friend class FacBom [friend]

Definition at line 19 of file YieldStore.hpp.

32.169.5.2 friend class FacBomManager [friend]

Definition at line 20 of file YieldStore.hpp.

32.169.6 Member Data Documentation

32.169.6.1 Key_T stdair::YieldStore::_key [protected]

The key of both structure and objects.

Definition at line 66 of file YieldStore.hpp.

Referenced by describeKey(), getAirlineCode(), getKey(), and toString().

32.169.6.2 BomAbstract* stdair::YieldStore::_parent [protected]

Definition at line 67 of file YieldStore.hpp.

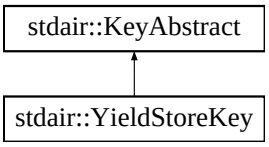
Referenced by getParent().

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

- stdair/bom/YieldStore.hpp
- stdair/bom/YieldStore.cpp

32.170 stdair::YieldStoreKey Struct Reference

`#include <stdair/bom/YieldStoreKey.hpp>`Inheritance diagram for stdair::YieldStoreKey::



Public Member Functions

- YieldStoreKey (const AirlineCode_T &iAirlineCode)
- YieldStoreKey (const YieldStoreKey &)
- ~YieldStoreKey ()
- const AirlineCode_T & getAirlineCode () const
- void toStream (std::ostream &ioOut) const
- void fromStream (std::istream &ioIn)
- const std::string toString () const

32.170.1 Detailed Description

Key of YieldStore.

Definition at line 14 of file YieldStoreKey.hpp.

32.170.2 Constructor & Destructor Documentation

32.170.2.1 stdair::YieldStoreKey::YieldStoreKey (const AirlineCode_T & *iAirlineCode*)

Constructors.

Definition at line 10 of file YieldStoreKey.cpp.

32.170.2.2 stdair::YieldStoreKey::YieldStoreKey (const YieldStoreKey & *iKey*)

Definition at line 14 of file YieldStoreKey.cpp.

32.170.2.3 stdair::YieldStoreKey::~YieldStoreKey ()

Destructor.

Definition at line 19 of file YieldStoreKey.cpp.

32.170.3 Member Function Documentation

32.170.3.1 const AirlineCode_T& stdair::YieldStoreKey::getAirlineCode () const [inline]

Get the airline code.

Definition at line 30 of file YieldStoreKey.hpp.

Referenced by stdair::YieldStore::getAirlineCode().

32.170.3.2 void stdair::YieldStoreKey::toStream (std::ostream & *ioOut*) const [virtual]

Dump a Business Object Key into an output stream.

Parameters:

ostream& the output stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 23 of file YieldStoreKey.cpp.

References toString().

32.170.3.3 void stdair::YieldStoreKey::fromStream (std::istream & *ioIn*) [virtual]

Read a Business Object Key from an input stream.

Parameters:

istream& the input stream.

Reimplemented from stdair::KeyAbstract.

Definition at line 28 of file YieldStoreKey.cpp.

32.170.3.4 const std::string stdair::YieldStoreKey::toString () const [virtual]

Get the serialised version of the Business Object Key.

That string is unique, at the level of a given Business Object, when among children of a given parent Business Object.

For instance, "H" and "K" allow to differentiate among two marketing classes for the same segment-date.

Reimplemented from stdair::KeyAbstract.

Definition at line 32 of file YieldStoreKey.cpp.

Referenced by stdair::YieldStore::describeKey(), toStream(), and stdair::YieldStore::toString().

The documentation for this struct was generated from the following files:

- stdair/bom/YieldStoreKey.hpp
- stdair/bom/YieldStoreKey.cpp

33 File Documentation

33.1 batches/stdair.cpp File Reference

33.2 batches/stdair.cpp

```
00001
00005 // STL
00006 #include <cassert>
00007 #include <iostream>
00008 #include <sstream>
00009 #include <fstream>
00010 #include <string>
00011 // Boost (Extended STL)
00012 #include <boost/date_time/posix_time/posix_time.hpp>
00013 #include <boost/date_time/gregorian/gregorian.hpp>
00014 #include <boost/program_options.hpp>
00015 #include <boost/tokenizer.hpp>
00016 #include <boost/lexical_cast.hpp>
00017 // StdAir
00018 #include <stdair/stdair_types.hpp>
00019 #include <stdair/bom/BomArchive.hpp>
00020 #include <stdair/bom/BookingRequestStruct.hpp>
00021 #include <stdair/bom/TravelSolutionStruct.hpp>
00022 #include <stdair/service/Logger.hpp>
00023 #include <stdair/STDAIR_Service.hpp>
00024 #include <stdair/config/stdair-paths.hpp>
00025
00026 // //////// Constants //////
00030 const std::string K_STDAIR_DEFAULT_LOG_FILENAME ("stdair.log");
00031
00035 const std::string K_STDAIR_DEFAULT_INPUT_FILENAME (STDAIR_SAMPLE_DIR
00036                                                     "/schedule01.csv");
00037
00042 const bool K_STDAIR_DEFAULT_BUILT_IN_INPUT = false;
00043
00049 const bool K_STDAIR_DEFAULT_BUILT_FOR_RMOL = false;
00050
00056 const bool K_STDAIR_DEFAULT_BUILT_FOR_CRS = false;
00057
00062 const int K_STDAIR_EARLY_RETURN_STATUS = 99;
00063
00064 // ///////// Parsing of Options & Configuration /////////
00065 // A helper function to simplify the main part.
00066 template<class T> std::ostream& operator<< (std::ostream& os,
00067                                             const std::vector<T>& v) {
00068   std::copy (v.begin(), v.end(), std::ostream_iterator<T> (std::cout, " "));
00069   return os;
00070 }
00071
00073 int readConfiguration (int argc, char* argv[], bool& ioIsBuiltin,
00074                        bool& ioIsForRMOL, bool& ioIsForCRS,
00075                        stdair::Filename_T& ioInputFilename,
00076                        std::string& ioLogFilename) {
00077   // Default for the built-in input
00078   ioIsBuiltin = K_STDAIR_DEFAULT_BUILT_IN_INPUT;
00079
00080   // Default for the RMOL input
00081   ioIsForRMOL = K_STDAIR_DEFAULT_BUILT_FOR_RMOL;
00082
00083   // Default for the CRS input
00084   ioIsForCRS = K_STDAIR_DEFAULT_BUILT_FOR_CRS;
00085
00086   // Declare a group of options that will be allowed only on command line
00087   boost::program_options::options_description generic ("Generic options");
00088   generic.add_options()
00089     ("prefix", "print installation prefix")
00090     ("version,v", "print version string")
00091     ("help,h", "produce help message");
00092
00093   // Declare a group of options that will be allowed both on command
```

```
  // line and in config file

  boost::program_options::options_description config ("Configuration");
  config.add_options()
    ("builtin,b",
     "The sample BOM tree can be either built-in or parsed from an input file. Th
at latter must then be given with the -i/--input option")
    ("rmol,r",
     "Build a sample BOM tree for RMOL (i.e., a dummy flight-date with a single l
eg-cabin)")
    ("crs,c",
     "Build a sample BOM tree for CRS")
    ("input,i",
     boost::program_options::value< std::string >(&ioInputFilename)->default_valu
e(K_STDAIR_DEFAULT_INPUT_FILENAME),
     "(CVS) input file for the demand distributions")
    ("log,l",
     boost::program_options::value< std::string >(&ioLogFilename)->default_value(
K_STDAIR_DEFAULT_LOG_FILENAME),
     "Filename for the logs")
    ;

  // Hidden options, will be allowed both on command line and
  // in config file, but will not be shown to the user.
  boost::program_options::options_description hidden ("Hidden options");
  hidden.add_options()
    ("copyright",
     boost::program_options::value< std::vector<std::string> >(),
     "Show the copyright (license)");

  boost::program_options::options_description cmdline_options;
  cmdline_options.add(generic).add(config).add(hidden);

  boost::program_options::options_description config_file_options;
  config_file_options.add(config).add(hidden);
  boost::program_options::options_description visible ("Allowed options");
  visible.add(generic).add(config);

  boost::program_options::positional_options_description p;
  p.add ("copyright", -1);

  boost::program_options::variables_map vm;
  boost::program_options::
    store (boost::program_options::command_line_parser (argc, argv).
           options (cmdline_options).positional(p).run(), vm);

  std::ifstream ifs ("stdair.cfg");
  boost::program_options::store (parse_config_file (ifs, config_file_options),
                                 vm);
  boost::program_options::notify (vm);

  if (vm.count ("help")) {
    std::cout << visible << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("version")) {
    std::cout << PACKAGE_NAME << ", version " << PACKAGE_VERSION << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("prefix")) {
    std::cout << "Installation prefix: " << PREFIXDIR << std::endl;
    return K_STDAIR_EARLY_RETURN_STATUS;
  }

  if (vm.count ("builtin")) {
```

```
    ioIsBuiltin = true;
  }

  if (vm.count ("rmol")) {
    ioIsForRMOL = true;

    // The RMOL sample tree takes precedence over the default built-in BOM tree
    ioIsBuiltin = false;
  }

  if (vm.count ("crs")) {
    ioIsForCRS = true;

    // The RMOL sample tree takes precedence over the default built-in BOM tree
    ioIsBuiltin = false;
  }

  const std::string isBuiltinStr = (ioIsBuiltin == true)?"yes":"no";
  std::cout << "The BOM should be built-in? " << isBuiltinStr << std::endl;

  const std::string isForRMOLStr = (ioIsForRMOL == true)?"yes":"no";
  std::cout << "The BOM should be built-in for RMOL? " << isForRMOLStr
            << std::endl;

  const std::string isForCRSStr = (ioIsForCRS == true)?"yes":"no";
  std::cout << "The BOM should be built-in for CRS? " << isForCRSStr
            << std::endl;

  if (ioIsBuiltin == false && ioIsForRMOL == false && ioIsForCRS == false) {
    if (vm.count ("input")) {
      ioInputFilename = vm["input"].as< std::string >();
      std::cout << "Input filename is: " << ioInputFilename << std::endl;

    } else {
      std::cerr << "Either one among the -b/--builtin, -r/--rmol, -c/--crs "
                << "or -i/--input options must be specified" << std::endl;
    }
  }

  if (vm.count ("log")) {
    ioLogFilename = vm["log"].as< std::string >();
    std::cout << "Log filename is: " << ioLogFilename << std::endl;
  }

  return 0;
}


// ///////////////// M A I N ////////////////////
int main (int argc, char* argv[]) {

  // State whether the BOM tree should be built-in or parsed from an
  // input file
  bool isBuiltin;

  // State whether a sample BOM tree should be built for RMOL.
  bool isForRMOL;

  // State whether a sample BOM tree should be built for the CRS.
  bool isForCRS;

  // Input file name
  stdair::Filename_T lInputFilename;

  // Output log File
  std::string lLogFilename;

```

```
00224   // Call the command-line option parser
00225   const int lOptionParserStatus =
00226     readConfiguration (argc, argv, isBuiltin, isForRMOL, isForCRS,
00227                        lInputFilename, lLogFilename);
00228
00229   if (lOptionParserStatus == K_STDAIR_EARLY_RETURN_STATUS) {
00230     return 0;
00231   }
00232
00233   // Set the log parameters
00234   std::ofstream logOutputFile;
00235   // Open and clean the log outputfile
00236   logOutputFile.open (lLogFilename.c_str());
00237   logOutputFile.clear();
00238
00239   const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
00240   stdair::STDAIR_Service stdairService (lLogParams);
00241
00242   // DEBUG
00243   STDAIR_LOG_DEBUG ("Welcome to stdair");
00244
00245   // Check wether or not a (CSV) input file should be read
00246   if (isBuiltin == true || isForRMOL == true || isForCRS == true) {
00247
00248     if (isForRMOL == true) {
00249       // Build the sample BOM tree for RMOL
00250       stdairService.buildDummyInventory (300);
00251
00252     } else if (isForCRS == true) {
00253       //
00254       stdair::TravelSolutionList_T lTravelSolutionList;
00255       stdairService.buildSampleTravelSolutions (lTravelSolutionList);
00256
00257       // Build the sample BOM tree for CRS
00258       const stdair::BookingRequestStruct& lBookingRequest =
00259         stdairService.buildSampleBookingRequest();
00260
00261       // DEBUG: Display the travel solution and booking request
00262       STDAIR_LOG_DEBUG ("Booking request: " << lBookingRequest.display());
00263
00264       const std::string& lCSVDump =
00265         stdairService.csvDisplay (lTravelSolutionList);
00266       STDAIR_LOG_DEBUG (lCSVDump);
00267
00268     } else {
00269       assert (isBuiltin == true);
00270
00271       // Build a sample BOM tree
00272       stdairService.buildSampleBom();
00273     }
00274
00275   } else {
00276     // Read the input file
00277     //stdairService.readFromInputFile (lInputFilename);
00278
00279     // DEBUG
00280     STDAIR_LOG_DEBUG ("StdAir will parse " << lInputFilename
00281                       << " and build the corresponding BOM tree.");
00282   }
00283
00284   // DEBUG: Display the whole persistent BOM tree
00285   const std::string& lCSVDump = stdairService.csvDisplay ();
00286   STDAIR_LOG_DEBUG (lCSVDump);
00287
00288   // Close the Log outputFile
00289   logOutputFile.close();
00290
```

```
  /*
    Note: as that program is not intended to be run on a server in
    production, it is better not to catch the exceptions. When it
    happens (that an exception is throwned), that way we get the
    call stack.
  */

  return 0;
}

```

33.3 doc/local/authors.doc File Reference

33.4 doc/local/codingrules.doc File Reference

33.5 doc/local/copyright.doc File Reference

33.6 doc/local/documentation.doc File Reference

33.7 doc/local/features.doc File Reference

33.8 doc/local/help_wanted.doc File Reference

33.9 doc/local/howto_release.doc File Reference

33.10 doc/local/index.doc File Reference

33.11 doc/local/installation.doc File Reference

33.12 doc/local/linking.doc File Reference

33.13 doc/local/test.doc File Reference

33.14 doc/local/users_guide.doc File Reference

33.15 doc/local/verification.doc File Reference

33.16 doc/tutorial/tutorial.doc File Reference

33.17 stdair/basic/BasChronometer.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasChronometer.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.18 stdair/basic/BasChronometer.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// Stdair
#include <stdair/basic/BasChronometer.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BasChronometer::BasChronometer () : _startTimeLaunched (false) {
  }

  // ////////////////////////////////////////////////////////////////////
  void BasChronometer::start () {
    // Get the time-stamp of now, and store it for later use
    _startTime = boost::posix_time::microsec_clock::local_time();

    // Update the boolean which states whether the chronometer
    // is launched
    _startTimeLaunched = true;
  }

  // ////////////////////////////////////////////////////////////////////
  double BasChronometer::elapsed () const {
    assert (_startTimeLaunched == true);

    // Get the time-stamp of now
    const boost::posix_time::ptime lStopTime =
      boost::posix_time::microsec_clock::local_time();

    // Calculate the time elapsed since the last time-stamp
    const boost::posix_time::time_duration lElapsedTime =
      lStopTime - _startTime;

    // Derived the corresponding number of milliseconds
    const double lElapsedTimeInMicroSeconds =
      static_cast<const double> (lElapsedTime.total_microseconds());

    // The elapsed time given in return is expressed in seconds
    return (lElapsedTimeInMicroSeconds / 1e6);
  }

}
```

33.19 stdair/basic/BasChronometer.hpp File Reference

```
#include <boost/date_time/posix_time/posix_time.hpp>
```

Classes

- struct stdair::BasChronometer

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.20 stdair/basic/BasChronometer.hpp

```
#ifndef __STDAIR_BAS_BASCHRONOMETER_HPP
#define __STDAIR_BAS_BASCHRONOMETER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// Boost (STL Extension)
// Boost Date-Time (http://boost.org/doc/html/date_time/posix_time.html)
#include <boost/date_time/posix_time/posix_time.hpp>

namespace stdair {

  struct BasChronometer {
    BasChronometer();

    void start ();

    std::string getStart () const {
      return boost::posix_time::to_simple_string (_startTime);
    }

    double elapsed () const;

  private:
    boost::posix_time::ptime _startTime;

    bool _startTimeLaunched;
  };

}
#endif // __STDAIR_BAS_BASCHRONOMETER_HPP
```

33.21 stdair/basic/BasConst.cpp File Reference

```
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/basic/BasConst_TravelSolution.hpp>
#include <stdair/basic/BasConst_SellUpCurves.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- const std::string stdair::DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ")
- const double stdair::DEFAULT_EPSILON_VALUE (0.0001)
- const unsigned int stdair::DEFAULT_FLIGHT_SPEED (900)
- const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES (0.0)
- const Duration_T stdair::NULL_BOOST_TIME_DURATION (-1,-1,-1)
- const Duration_T stdair::DEFAULT_NULL_DURATION (0, 0, 0)
- const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR (365)
- const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS (1000)
- const DayDuration_T stdair::DEFAULT_DAY_DURATION (0)
- const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD (Date_T(2007, 1, 1), Date_T(2007, 1, 1))
- const DOW_String_T stdair::DEFAULT_DOW_STRING ("0000000")
- const DateOffset_T stdair::DEFAULT_DATE_OFFSET (0)
- const Date_T stdair::DEFAULT_DATE (2010, boost::gregorian::Jan, 1)
- const DateTime_T stdair::DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION)
- const Duration_T stdair::DEFAULT_EPSILON_DURATION (0, 0, 0, 1)
- const Count_T stdair::SECONDS_IN_ONE_DAY (86400)
- const Count_T stdair::MILLISECONDS_IN_ONE_SECOND (1000)
- const RandomSeed_T stdair::DEFAULT_RANDOM_SEED (120765987)
- const AirportCode_T stdair::AIRPORT_LHR ("LHR")
- const AirportCode_T stdair::AIRPORT_SYD ("SYD")
- const CityCode_T stdair::POS_LHR ("LHR")
- const Date_T stdair::DATE_20110115 (2011, boost::gregorian::Jan, 15)

- const Date_T stdair::DATE_20111231 (2011, boost::gregorian::Dec, 31)
- const DayDuration_T stdair::NO_ADVANCE_PURCHASE (0)
- const SaturdayStay_T stdair::SATURDAY_STAY (true)
- const SaturdayStay_T stdair::NO_SATURDAY_STAY (false)
- const ChangeFees_T stdair::CHANGE_FEES (true)
- const ChangeFees_T stdair::NO_CHANGE_FEES (false)
- const NonRefundable_T stdair::NON_REFUNDABLE (true)
- const NonRefundable_T stdair::NO_NON_REFUNDABLE (false)
- const SaturdayStay_T stdair::DEFAULT_BOM_TREE_SATURDAY_STAY (true)
- const ChangeFees_T stdair::DEFAULT_BOM_TREE_CHANGE_FEES (true)
- const NonRefundable_T stdair::DEFAULT_BOM_TREE_NON_REFUNDABLE (true)
- const DayDuration_T stdair::NO_STAY_DURATION (0)
- const AirlineCode_T stdair::AIRLINE_CODE_BA ("BA")
- const CabinCode_T stdair::CABIN_Y ("Y")
- const ClassCode_T stdair::CLASS_CODE_Y ("Y")
- const ClassCode_T stdair::CLASS_CODE_Q ("Q")
- const AirportCode_T stdair::AIRPORT_SIN ("SIN")
- const AirportCode_T stdair::AIRPORT_BKK ("BKK")
- const CityCode_T stdair::POS_SIN ("SIN")
- const CabinCode_T stdair::CABIN_ECO ("Eco")
- const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER ("M")
- const FamilyCode_T stdair::DEFAULT_FAMILY_CODE ("0")
- const PolicyCode_T stdair::DEFAULT_POLICY_CODE ("0")
- const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT")
- const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")
- const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based Nesting")
- const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE ("0")
- const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES (0)
- const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE ("")
- const Distance_T stdair::DEFAULT_DISTANCE_VALUE (0)
- const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE ("CC")
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0)
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION (0)
- const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS (0)
- const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS (0)
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY (100.0)
- const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE (0.0)
- const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE (0.0)
- const Availability_T stdair::DEFAULT_NULL_AVAILABILITY (0.0)
- const Availability_T stdair::DEFAULT_AVAILABILITY (9.0)
- const Availability_T stdair::MAXIMAL_AVAILABILITY (9999.0)
- const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG (false)
- const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT (9999.0)
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0)

- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0)
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0)
- const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE (0.0)
- const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE (0.0)
- const Fare_T stdair::DEFAULT_FARE_VALUE (0.0)
- const Yield_T stdair::DEFAULT_CLASS_YIELD_VALUE (0.0)
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE (0.0)
- const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE (100.0)
- const Yield_T stdair::DEFAULT_YIELD_VALUE (0.0)
- const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE (std::numeric_limits< double >::max())
- const NbOfBookings_T stdair::DEFAULT_YIELD_NB_OF_BOOKINGS (0.0)
- const Identity_T stdair::DEFAULT_BOOKING_NUMBER (0)
- const NbOfCancellations_T stdair::DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0)
- const NbOfNoShows_T stdair::DEFAULT_YIELD_NB_OF_NOSHOWS (0.0)
- const Availability_T stdair::DEFAULT_YIELD_AVAILABILITY (0.0)
- const CensorshipFlag_T stdair::DEFAULT_YIELD_CENSORSHIPFLAG (false)
- const BookingLimit_T stdair::DEFAULT_YIELD_BOOKING_LIMIT (0.0)
- const OverbookingRate_T stdair::DEFAULT_YIELD_OVERBOOKING_RATE (0.0)
- const Fare_T stdair::DEFAULT_OND_FARE_VALUE (0.0)
- const Count_T stdair::DEFAULT_PROGRESS_STATUS (0)
- const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS (100)
- const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1)
- const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_-OLDEST_DATE, NULL_BOOST_TIME_DURATION)
- const PartySize_T stdair::DEFAULT_PARTY_SIZE (1)
- const DayDuration_T stdair::DEFAULT_STAY_DURATION (7)
- const WTP_T stdair::DEFAULT_WTP (1000.0)
- const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_-DATE)
- const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0)
- const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE (22)
- const Date_T stdair::DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_-DATE-DEFAULT_ADVANCE_PURCHASE)
- const Duration_T stdair::DEFAULT_REQUEST_TIME (8, 0, 0)
- const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE, DEFAULT_REQUEST_TIME)
- const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN ("M")
- const CityCode_T stdair::DEFAULT_POS ("ALL")
- const ChannelLabel_T stdair::DEFAULT_CHANNEL ("DC")
- const ChannelLabel_T stdair::CHANNEL_DN ("DN")
- const ChannelLabel_T stdair::CHANNEL_IN ("IN")
- const TripType_T stdair::TRIP_TYPE_ONE_WAY ("OW")
- const TripType_T stdair::TRIP_TYPE_ROUND_TRIP ("RT")
- const TripType_T stdair::TRIP_TYPE_INBOUND ("RI")
- const TripType_T stdair::TRIP_TYPE_OUTBOUND ("RO")
- const FrequentFlyer_T stdair::DEFAULT_FF_TIER ("N")
- const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME (100.0)
- const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS (3600)
- const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0)

- const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0)
- const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR (0.0)
- const PriceCurrency_T stdair::DEFAULT_CURRENCY ("EUR")
- const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS (false)
- const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE ("XX")
- const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE ("")
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER (9999)
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF (255)
- const TableID_T stdair::DEFAULT_TABLE_ID (9999)
- const Date_T stdair::DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1)
- const AirportCode_T stdair::DEFAULT_AIRPORT_CODE ("XXX")
- const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE ("")
- const AirportCode_T stdair::DEFAULT_ORIGIN ("XXX")
- const AirportCode_T stdair::DEFAULT_DESTINATION ("YYY")
- const CabinCode_T stdair::DEFAULT_CABIN_CODE ("X")
- const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE ("EcoSaver")
- const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF")
- const ClassCode_T stdair::DEFAULT_CLASS_CODE ("X")
- const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE ("")
- const BidPrice_T stdair::DEFAULT_BID_PRICE (0.0)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7)
- const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3)
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX (1)
- const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER (0)
- const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT (0)
- const UPR_T stdair::DEFAULT_NULL_UPR (0)
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF")
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC")
- const std::string stdair::DEFAULT_KEY_FLD_DELIMITER (";")
- const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER (",")
- const boost::char_separator< char > stdair::DEFAULT_KEY_TOKEN_DELIMITER (";, ")

Variables

- const std::string stdair::DOW_STR [ ]
- const UnconstrainingMethod stdair::DEFAULT_UNCONSTRAINING_METHOD ('E')
- const PartnershipTechnique stdair::DEFAULT_PARTNERSHIP_TECHNIQUE ('N')
- const ForecastingMethod stdair::DEFAULT_FORECASTING_METHOD ('Q')
- const PreOptimisationMethod stdair::DEFAULT_PREOPTIMISATION_METHOD ('N')
- const OptimisationMethod stdair::DEFAULT_OPTIMISATION_METHOD ('M')
- const CensorshipFlagList_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG_LIST
- const Date_T stdair::DEFAULT_DICO_STUDIED_DATE
- const AirlineCodeList_T stdair::DEFAULT_AIRLINE_CODE_LIST
- const ClassList_StringList_T stdair::DEFAULT_CLASS_CODE_LIST
- const BidPriceVector_T stdair::DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>()
- const int stdair::DEFAULT_MAX_DTD = 365
- const DCPList_T stdair::DEFAULT_DCP_LIST = DefaultDCPList::init()
- const FRAT5Curve_T stdair::FRAT5_CURVE_A
- const FRAT5Curve_T stdair::FRAT5_CURVE_B
- const FRAT5Curve_T stdair::FRAT5_CURVE_C

- const FRAT5Curve_T stdair::FRAT5_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_A
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_B
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_C
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_D
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_E
- const FFDisutilityCurve_T stdair::FF_DISUTILITY_CURVE_F
- const DTDFratMap_T stdair::DEFAULT_DTD_FRAT5COEF_MAP
- const DTDProbMap_T stdair::DEFAULT_DTD_PROB_MAP
- const OnDStringList_T stdair::DEFAULT_OND_STRING_LIST
- const std::string stdair::DISPLAY_LEVEL_STRING_ARRAY [51]

33.22 stdair/basic/BasConst.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // StdAir
00005 #include <stdair/basic/BasConst_General.hpp>
00006 #include <stdair/basic/BasConst_BomDisplay.hpp>
00007 #include <stdair/basic/BasConst_Event.hpp>
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_BookingClass.hpp>
00011 #include <stdair/basic/BasConst_Yield.hpp>
00012 #include <stdair/basic/BasConst_DefaultObject.hpp>
00013 #include <stdair/basic/BasConst_Period_BOM.hpp>
00014 #include <stdair/basic/BasConst_TravelSolution.hpp>
00015 #include <stdair/basic/BasConst_SellUpCurves.hpp>
00016
00017 namespace stdair {
00018
00019   // ///////// General /////////
00021   const std::string DEFAULT_BOM_ROOT_KEY (" -- ROOT -- ");
00022
00024   const double DEFAULT_EPSILON_VALUE (0.0001);
00025
00027   const unsigned int DEFAULT_FLIGHT_SPEED (900);
00028
00030   const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES (0.0);
00031
00033   const Duration_T NULL_BOOST_TIME_DURATION (-1, -1, -1);
00034
00036   const Duration_T DEFAULT_NULL_DURATION (0, 0, 0);
00037
00039   const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR (365);
00040
00042   const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS (1000);
00043
00044   // //////// (Flight-)Period-related BOM ///////
00046   const DayDuration_T DEFAULT_DAY_DURATION (0);
00047
00049   const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD (Date_T (2007, 1, 1),
00050                                                 Date_T (2007, 1, 1));
00051
00053   const std::string DOW_STR[] =
00054     {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"};
00055
00057   const DOW_String_T DEFAULT_DOW_STRING ("0000000");
00058
00060   const DateOffset_T DEFAULT_DATE_OFFSET (0);
00061
00062
00063   //  // //////// General ///////
00065   const Date_T DEFAULT_DATE (2010, boost::gregorian::Jan, 1);
00066
00068   const DateTime_T DEFAULT_DATETIME (DEFAULT_DATE, NULL_BOOST_TIME_DURATION);
00069
00071   const Duration_T DEFAULT_EPSILON_DURATION (0, 0, 0, 1);
00072
00074   const Count_T SECONDS_IN_ONE_DAY (86400);
00075
00077   const Count_T MILLISECONDS_IN_ONE_SECOND (1000);
00078
00080   const RandomSeed_T DEFAULT_RANDOM_SEED (120765987);
00081
00082   // //////// Default BOM tree objects /////// ///////
00084   const AirportCode_T AIRPORT_LHR ("LHR");
00085
```

```
00087   const AirportCode_T AIRPORT_SYD ("SYD");
00088
00090   const CityCode_T POS_LHR ("LHR");
00091
00093   const Date_T DATE_20110115 (2011, boost::gregorian::Jan, 15);
00094   const Date_T DATE_20111231 (2011, boost::gregorian::Dec, 31);
00095
00097   const DayDuration_T NO_ADVANCE_PURCHASE (0);
00098
00100   const SaturdayStay_T SATURDAY_STAY (true);
00101
00103   const SaturdayStay_T NO_SATURDAY_STAY (false);
00104
00106   const ChangeFees_T CHANGE_FEES (true);
00107
00109   const ChangeFees_T NO_CHANGE_FEES (false);
00110
00112   const NonRefundable_T NON_REFUNDABLE (true);
00113
00115   const NonRefundable_T NO_NON_REFUNDABLE (false);
00116
00118   const SaturdayStay_T DEFAULT_BOM_TREE_SATURDAY_STAY (true);
00119
00121   const ChangeFees_T DEFAULT_BOM_TREE_CHANGE_FEES (true);
00122
00124   const NonRefundable_T DEFAULT_BOM_TREE_NON_REFUNDABLE (true);
00125
00127   const DayDuration_T NO_STAY_DURATION (0);
00128
00130   const AirlineCode_T AIRLINE_CODE_BA ("BA");
00131
00133   const CabinCode_T  CABIN_Y ("Y");
00134
00136   const ClassCode_T  CLASS_CODE_Y ("Y");
00137
00138   // //////// Travel solutions related objects///////
00140   const ClassCode_T CLASS_CODE_Q ("Q");
00141
00142   // //////// Booking request related objects///////
00144   const AirportCode_T AIRPORT_SIN ("SIN");
00145
00147   const AirportCode_T AIRPORT_BKK ("BKK");
00148
00150   const CityCode_T POS_SIN ("SIN");
00151
00153   const CabinCode_T CABIN_ECO ("Eco");
00154
00156   const FrequentFlyer_T FREQUENT_FLYER_MEMBER ("M");
00157
00158   // //////// Default ///////
00160   const FamilyCode_T DEFAULT_FAMILY_CODE ("0");
00161
00163   const PolicyCode_T DEFAULT_POLICY_CODE ("0");
00164
00166   const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE ("DEFAULT");
00167
00169   const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE ("Display Nesting")
      ;
00170
00172   const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE ("Yield-Based N
      esting");
00173
00175   const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE ("0");
00176
00178   const NbOfAirlines_T DEFAULT_NBOFAIRLINES (0);
00179
00181   const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE ("");
```

```
00182
00183   // //////// Booking-class-related BOM ///////
00185   const Distance_T DEFAULT_DISTANCE_VALUE (0);
00186
00188   const ClassCode_T DEFAULT_CLOSED_CLASS_CODE ("CC");
00189
00192   const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS (0);
00193
00196   const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS (0);
00197
00199   const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND (0);
00200
00202   const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN (0);
00203
00205   const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION (0);
00206
00208   const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS (0);
00209
00211   const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS (0);
00212
00214   const CabinCapacity_T DEFAULT_CABIN_CAPACITY (100.0);
00215
00217   const CommittedSpace_T DEFAULT_COMMITTED_SPACE (0.0);
00218
00220   const BlockSpace_T DEFAULT_BLOCK_SPACE (0.0);
00221
00223   const Availability_T DEFAULT_NULL_AVAILABILITY (0.0);
00224
00226   const Availability_T DEFAULT_AVAILABILITY (9.0);
00227
00229   const Availability_T MAXIMAL_AVAILABILITY (9999.0);
00230
00232   const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD ('E');
00233
00235   const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE ('N');
00236
00238   const ForecastingMethod DEFAULT_FORECASTING_METHOD ('Q');
00239
00241   const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD ('N');
00242
00244   const OptimisationMethod DEFAULT_OPTIMISATION_METHOD ('M');;
00245
00246   // //////// (Segment-)Class-related BOM ///////
00249   const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG (false);
00250
00253   const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST =
00254     std::vector<CensorshipFlag_T>();
00255
00257   const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT (9999.0);
00258
00260   const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL (9999.0);
00261
00263   const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL (9999.0);
00264
00266   const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL (0.0);
00267
00269   const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE (0.0);
00270
00272   const BookingRatio_T DEFAULT_OND_BOOKING_RATE (0.0);
00273
00275   const Fare_T DEFAULT_FARE_VALUE (0.0);
00276
00278   const Yield_T DEFAULT_CLASS_YIELD_VALUE (0.0);
00279
00281   const Revenue_T DEFAULT_REVENUE_VALUE (0.0);
00282
00284   const Percentage_T DEFAULT_LOAD_FACTOR_VALUE (100.0);
```

```
00285
00286
00287   // //////// (Leg-)YieldRange-related BOM ///////
00289   const Yield_T DEFAULT_YIELD_VALUE (0.0);
00290
00292   const Yield_T DEFAULT_YIELD_MAX_VALUE (std::numeric_limits<double>::max());
00293
00295   const NbOfBookings_T DEFAULT_YIELD_NB_OF_BOOKINGS (0.0);
00296
00298   const Identity_T DEFAULT_BOOKING_NUMBER (0);
00299
00301   const NbOfCancellations_T DEFAULT_YIELD_NB_OF_CANCELLATIONS (0.0);
00302
00304   const NbOfNoShows_T DEFAULT_YIELD_NB_OF_NOSHOWS (0.0);
00305
00307   const Availability_T DEFAULT_YIELD_AVAILABILITY (0.0);
00308
00311   const CensorshipFlag_T DEFAULT_YIELD_CENSORSHIPFLAG (false);
00312
00314   const BookingLimit_T DEFAULT_YIELD_BOOKING_LIMIT (0.0);
00315
00317   const OverbookingRate_T DEFAULT_YIELD_OVERBOOKING_RATE (0.0);
00318
00319
00320   // //////// OnD-related BOM ///////
00322   const Fare_T DEFAULT_OND_FARE_VALUE (0.0);
00323
00324
00325   // //////// Event Generation /////////
00326
00328   const Count_T DEFAULT_PROGRESS_STATUS (0);
00329
00331   const Percentage_T MAXIMUM_PROGRESS_STATUS (100);
00332
00335   const Date_T DEFAULT_EVENT_OLDEST_DATE (2008, boost::gregorian::Jan, 1);
00336
00339   const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME (DEFAULT_EVENT_OLDEST_DATE,
00340                                                   NULL_BOOST_TIME_DURATION);
00341
00342
00343   // //////// Booking Request /////////
00345   const PartySize_T DEFAULT_PARTY_SIZE (1);
00346
00348   const DayDuration_T DEFAULT_STAY_DURATION (7);
00349
00351   const WTP_T DEFAULT_WTP (1000.0);
00352
00354   const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE (DEFAULT_DEPARTURE_DATE);
00355
00357   const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME (8, 0, 0);
00358
00360   const DateOffset_T DEFAULT_ADVANCE_PURCHASE (22);
00361
00363   const Date_T DEFAULT_REQUEST_DATE (DEFAULT_PREFERRED_DEPARTURE_DATE
00364                                      - DEFAULT_ADVANCE_PURCHASE);
00365
00367   const Duration_T DEFAULT_REQUEST_TIME (8, 0, 0);
00368
00370   const DateTime_T DEFAULT_REQUEST_DATE_TIME (DEFAULT_REQUEST_DATE,
00371                                               DEFAULT_REQUEST_TIME);
00372
00374   const CabinCode_T DEFAULT_PREFERRED_CABIN ("M");
00375
00377   const CityCode_T DEFAULT_POS ("ALL");
00378
00380   const ChannelLabel_T DEFAULT_CHANNEL ("DC");
00381
```

```
00383   const ChannelLabel_T CHANNEL_DN ("DN");
00384
00386   const ChannelLabel_T CHANNEL_IN ("IN");
00387
00389   const TripType_T TRIP_TYPE_ONE_WAY ("OW");
00390
00392   const TripType_T TRIP_TYPE_ROUND_TRIP ("RT");
00393
00395   const TripType_T TRIP_TYPE_INBOUND ("RI");
00396
00398   const TripType_T TRIP_TYPE_OUTBOUND ("RO");
00399
00401   const FrequentFlyer_T DEFAULT_FF_TIER ("N");
00402
00404   const PriceValue_T DEFAULT_VALUE_OF_TIME (100.0);
00405
00407   const IntDuration_T HOUR_CONVERTED_IN_SECONDS (3600);
00408
00409   // //////// Travel Solutions ///////
00411   const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME (0, 30, 0);
00412
00414   const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME (24, 0, 0);
00415
00417   const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR (0.0);
00418
00420   const PriceCurrency_T DEFAULT_CURRENCY ("EUR");
00421
00423   const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS (false);
00424
00426   const Date_T DEFAULT_DICO_STUDIED_DATE;
00427
00428   // //////// Inventory-related BOM ///////
00430   const AirlineCode_T DEFAULT_AIRLINE_CODE ("XX");
00431
00433   const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE ("");
00434
00436   const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;
00437
00439   const FlightNumber_T DEFAULT_FLIGHT_NUMBER (9999);
00440
00442   const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF (255);
00443
00445   const TableID_T DEFAULT_TABLE_ID (9999);
00446
00448   const Date_T DEFAULT_DEPARTURE_DATE (1900, boost::gregorian::Jan, 1);
00449
00451   const AirportCode_T DEFAULT_AIRPORT_CODE ("XXX");
00452
00454   const AirportCode_T DEFAULT_NULL_AIRPORT_CODE ("");
00455
00457   const AirportCode_T DEFAULT_ORIGIN ("XXX");
00458
00460   const AirportCode_T DEFAULT_DESTINATION ("YYY");
00461
00463   const CabinCode_T DEFAULT_CABIN_CODE ("X");
00464
00466   const FamilyCode_T DEFAULT_FARE_FAMILY_CODE ("EcoSaver");
00467
00469   const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE ("NoFF");
00470
00472   const ClassCode_T DEFAULT_CLASS_CODE ("X");
00473
00475   const ClassCode_T DEFAULT_NULL_CLASS_CODE ("");
00476
00478   const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST;
00479
00481   const BidPrice_T DEFAULT_BID_PRICE (0.0);
```

```
00482
00484   const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR = std::vector<BidPrice_T>();
00485
00489   const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT (7);
00490
00493   const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND (3);
00494
00496   const SeatIndex_T DEFAULT_SEAT_INDEX (1);
00497
00499   const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER (0);
00500
00502   const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT (0);
00503
00505   const UPR_T DEFAULT_NULL_UPR (0);
00506
00508   const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE ("FF");
00509
00511   const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE ("SC");
00512
00514   const int DEFAULT_MAX_DTD = 365;
00515
00517   const DCPList_T DEFAULT_DCP_LIST = DefaultDCPList::init();
00518   DCPList_T DefaultDCPList::init() {
00519     DCPList_T oDCPList;
00520     //oDCPList.push_back (72);
00521     oDCPList.push_back (63);oDCPList.push_back (56);oDCPList.push_back (49);
00522     oDCPList.push_back (42);oDCPList.push_back (35);oDCPList.push_back (31);
00523     oDCPList.push_back (27);oDCPList.push_back (23);oDCPList.push_back (19);
00524     oDCPList.push_back (16);oDCPList.push_back (13);oDCPList.push_back (10);
00525     oDCPList.push_back (7); oDCPList.push_back (5); oDCPList.push_back (3);
00526     oDCPList.push_back (1); oDCPList.push_back (0);
00527     return oDCPList;
00528   }
00529
00531   const FRAT5Curve_T FRAT5_CURVE_A =
00532     DefaultMap::createFRAT5CurveA();
00533   FRAT5Curve_T DefaultMap::createFRAT5CurveA() {
00534     FRAT5Curve_T oCurve;
00535     oCurve[63] = 1.05;  oCurve[56] = 1.07;  oCurve[49] = 1.09;
00536     oCurve[42] = 1.11;  oCurve[35] = 1.14;  oCurve[31] = 1.16;
00537     oCurve[27] = 1.18;  oCurve[23] = 1.21;  oCurve[19] = 1.24;
00538     oCurve[16] = 1.27;  oCurve[13] = 1.30;  oCurve[10] = 1.33;
00539     oCurve[7]  = 1.37;  oCurve[5]  = 1.40;  oCurve[3]  = 1.45;
00540     oCurve[1]  = 1.50;
00541     return oCurve;
00542   }
00543
00545   const FRAT5Curve_T FRAT5_CURVE_B =
00546     DefaultMap::createFRAT5CurveB();
00547   FRAT5Curve_T DefaultMap::createFRAT5CurveB() {
00548     FRAT5Curve_T oCurve;
00549     oCurve[63] = 1.20;  oCurve[56] = 1.23;  oCurve[49] = 1.26;
00550     oCurve[42] = 1.30;  oCurve[35] = 1.35;  oCurve[31] = 1.40;
00551     oCurve[27] = 1.50;  oCurve[23] = 1.60;  oCurve[19] = 1.80;
00552     oCurve[16] = 2.10;  oCurve[13] = 2.20;  oCurve[10] = 2.30;
00553     oCurve[7]  = 2.40;  oCurve[5]  = 2.44;  oCurve[3]  = 2.47;
00554     oCurve[1]  = 2.50;
00555     return oCurve;
00556   }
00557
00559   const FRAT5Curve_T FRAT5_CURVE_C =
00560     DefaultMap::createFRAT5CurveC();
00561   FRAT5Curve_T DefaultMap::createFRAT5CurveC() {
00562     FRAT5Curve_T oCurve;
00563     oCurve[63] = 1.40;  oCurve[56] = 1.45;  oCurve[49] = 1.50;
00564     oCurve[42] = 1.55;  oCurve[35] = 1.60;  oCurve[31] = 1.70;
00565     oCurve[27] = 1.80;  oCurve[23] = 2.00;  oCurve[19] = 2.30;
```

```
    oCurve[16] = 2.60;  oCurve[13] = 3.00;  oCurve[10] = 3.30;
    oCurve[7]  = 3.40;  oCurve[5]  = 3.44;  oCurve[3]  = 3.47;
    oCurve[1]  = 3.50;
    return oCurve;
  }

  const FRAT5Curve_T FRAT5_CURVE_D =
    DefaultMap::createFRAT5CurveD();
  FRAT5Curve_T DefaultMap::createFRAT5CurveD() {
    FRAT5Curve_T oCurve;
    oCurve[63] = 1.60;  oCurve[56] = 1.67;  oCurve[49] = 1.74;
    oCurve[42] = 1.81;  oCurve[35] = 1.88;  oCurve[31] = 2.00;
    oCurve[27] = 2.15;  oCurve[23] = 2.45;  oCurve[19] = 2.75;
    oCurve[16] = 3.20;  oCurve[13] = 3.80;  oCurve[10] = 4.25;
    oCurve[7]  = 4.35;  oCurve[5]  = 4.40;  oCurve[3]  = 4.45;
    oCurve[1]  = 4.50;
    return oCurve;
  }

  const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A =
    DefaultMap::createFFDisutilityCurveA();
  FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveA() {
    FFDisutilityCurve_T oCurve;
    oCurve[63] = 0.0098;  oCurve[56] = 0.0096;  oCurve[49] = 0.0093;
    oCurve[42] = 0.0090;  oCurve[35] = 0.0086;  oCurve[31] = 0.0082;
    oCurve[27] = 0.0077;  oCurve[23] = 0.0071;  oCurve[19] = 0.0065;
    oCurve[16] = 0.0059;  oCurve[13] = 0.0052;  oCurve[10] = 0.0045;
    oCurve[7]  = 0.0039;  oCurve[5]  = 0.0036;  oCurve[3]  = 0.0033;
    oCurve[1]  = 0.0030;
    return oCurve;
  }

  const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B =
    DefaultMap::createFFDisutilityCurveB();
  FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveB() {
    FFDisutilityCurve_T oCurve;
    oCurve[63] = 0.0082;  oCurve[56] = 0.0080;  oCurve[49] = 0.0078;
    oCurve[42] = 0.0075;  oCurve[35] = 0.0072;  oCurve[31] = 0.0068;
    oCurve[27] = 0.0064;  oCurve[23] = 0.0059;  oCurve[19] = 0.0054;
    oCurve[16] = 0.0049;  oCurve[13] = 0.0044;  oCurve[10] = 0.0038;
    oCurve[7]  = 0.0033;  oCurve[5]  = 0.0030;  oCurve[3]  = 0.0028;
    oCurve[1]  = 0.0025;
    return oCurve;
  }

  const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C =
    DefaultMap::createFFDisutilityCurveC();
  FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveC() {
    FFDisutilityCurve_T oCurve;
    oCurve[63] = 0.0065;  oCurve[56] = 0.0064;  oCurve[49] = 0.0062;
    oCurve[42] = 0.0060;  oCurve[35] = 0.0057;  oCurve[31] = 0.0054;
    oCurve[27] = 0.0051;  oCurve[23] = 0.0047;  oCurve[19] = 0.0043;
    oCurve[16] = 0.0039;  oCurve[13] = 0.0035;  oCurve[10] = 0.0030;
    oCurve[7]  = 0.0026;  oCurve[5]  = 0.0024;  oCurve[3]  = 0.0022;
    oCurve[1]  = 0.0020;
    return oCurve;
  }

  const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D =
    DefaultMap::createFFDisutilityCurveD();
  FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveD() {
    FFDisutilityCurve_T oCurve;
    oCurve[63] = 0.0050;  oCurve[56] = 0.0049;  oCurve[49] = 0.0047;
    oCurve[42] = 0.0045;  oCurve[35] = 0.0043;  oCurve[31] = 0.0040;
    oCurve[27] = 0.0037;  oCurve[23] = 0.0034;  oCurve[19] = 0.0030;
    oCurve[16] = 0.0026;  oCurve[13] = 0.0022;  oCurve[10] = 0.0017;
    oCurve[7]  = 0.0013;  oCurve[5]  = 0.0012;  oCurve[3]  = 0.0011;
```

```
00654     oCurve[1]  = 0.0010;
00655     return oCurve;
00656   }
00657
00663   const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E =
00664     DefaultMap::createFFDisutilityCurveE();
00665   FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveE() {
00666     FFDisutilityCurve_T oCurve;
00667     oCurve[63] = 0.0043;  oCurve[56] = 0.0042;  oCurve[49] = 0.0041;
00668     oCurve[42] = 0.0039;  oCurve[35] = 0.0037;  oCurve[31] = 0.0035;
00669     oCurve[27] = 0.0032;  oCurve[23] = 0.0029;  oCurve[19] = 0.0025;
00670     oCurve[16] = 0.0021;  oCurve[13] = 0.0018;  oCurve[10] = 0.0013;
00671     oCurve[7]  = 0.0011;  oCurve[5]  = 0.0010;  oCurve[3]  = 0.0009;
00672     oCurve[1]  = 0.0008;
00673     return oCurve;
00674   }
00675
00681   const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F =
00682     DefaultMap::createFFDisutilityCurveF();
00683   FFDisutilityCurve_T DefaultMap::createFFDisutilityCurveF() {
00684     FFDisutilityCurve_T oCurve;
00685     oCurve[63] = 0.0032;  oCurve[56] = 0.0031;  oCurve[49] = 0.0030;
00686     oCurve[42] = 0.0029;  oCurve[35] = 0.0027;  oCurve[31] = 0.0025;
00687     oCurve[27] = 0.0022;  oCurve[23] = 0.0019;  oCurve[19] = 0.0016;
00688     oCurve[16] = 0.0013;  oCurve[13] = 0.0010;  oCurve[10] = 0.0008;
00689     oCurve[7]  = 0.0007;  oCurve[5]  = 0.0006;  oCurve[3]  = 0.0005;
00690     oCurve[1]  = 0.0004;
00691     return oCurve;
00692   }
00693
00695   const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP =
00696     DefaultDtdFratMap::init();
00697   DTDFratMap_T DefaultDtdFratMap::init() {
00698     DTDFratMap_T oDFCMap;
00699     oDFCMap[71] = 2.50583571429; oDFCMap[63] = 2.55994571429;
00700     oDFCMap[56] = 2.60841857143; oDFCMap[49] = 2.68888;
00701     oDFCMap[42] = 2.78583714286; oDFCMap[35] = 2.89091428571;
00702     oDFCMap[31] = 2.97871428571; oDFCMap[28] = 3.05521428571;
00703     oDFCMap[24] = 3.15177142857; oDFCMap[21] = 3.22164285714;
00704     oDFCMap[17] = 3.32237142857; oDFCMap[14] = 3.38697142857;
00705     oDFCMap[10] = 3.44204285714; oDFCMap[7] = 3.46202857143;
00706     oDFCMap[5] = 3.47177142857;  oDFCMap[3] = 3.4792;
00707     oDFCMap[1] = 3.48947142857; // oDFCMap[0] = 3.49111428571;
00708     return oDFCMap;
00709   }
00710
00712   const DTDProbMap_T DEFAULT_DTD_PROB_MAP =
00713     DefaultDtdProbMap::init();
00714   DTDProbMap_T DefaultDtdProbMap::init() {
00715     DTDProbMap_T oDPMap;
00716     oDPMap[-330] = 0; oDPMap[-150] = 0.1; oDPMap[-92] = 0.2;
00717     oDPMap[-55] = 0.3; oDPMap[-34] = 0.4; oDPMap[-21] = 0.5;
00718     oDPMap[-12] = 0.6; oDPMap[-6] = 0.7; oDPMap[-3] = 0.8;
00719     oDPMap[-1] = 0.9; oDPMap[0] = 1.0;
00720     return oDPMap;
00721   }
00722
00723   // ////////// Key and display related /////////////
00726   const std::string DEFAULT_KEY_FLD_DELIMITER (";");
00727
00730   const std::string DEFAULT_KEY_SUB_FLD_DELIMITER (",");
00731
00733   const boost::char_separator<char> DEFAULT_KEY_TOKEN_DELIMITER (";, ");
00734
00736   const OnDStringList_T DEFAULT_OND_STRING_LIST;
00737
00738
```

```
  // ////////// BomManager-related constants ///////////
  const std::string DISPLAY_LEVEL_STRING_ARRAY[51] =
  { "", "  ", "    ", "      ",
    "        ", "          ", "            ", "              ",
    "                ", "                  ", "                    ",
    "                      ", "                        ",
    "                          ", "                            ",
    "                              ", "                                ",
    "                                    ",
    "                                      ",
    "                                        ",
    "                                          ",
    "                                            ",
    "                                              ",
    "                                                ",
    "                                                  ",
    "                                                    ",
    "                                                      ",
    "                                                        ",
    "                                                          ",
    "                                                            ",
    "                                                              ",
    "                                                                ",
    "                                                                  ",
    "                                                                    ",
    "                                                                      ",
    "                                                                        ",
    "                                                                          ",
    "                                                                            ",
    "                                                                              ",
    "                                                                                ",
    "                                                                                  ",
    "                                                                                    ",
    "                                                                                      ",
    "                                                                                        ",
    "                                                                                          ",
    "                                                                                            ",
    "                                                                                              ",
    "                                                                                                ",
    "                                                                                                  ",
    "                                                                                                    ",
    "                                                                                                      " };


}
```

33.23 stdair/basic/BasConst_BomDisplay.hpp File Reference

```
#include <string>
#include <boost/tokenizer.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const std::string stdair::DEFAULT_KEY_FLD_DELIMITER
- const std::string stdair::DEFAULT_KEY_SUB_FLD_DELIMITER
- const boost::char_separator< char > stdair::DEFAULT_KEY_TOKEN_DELIMITER

33.24 stdair/basic/BasConst_BomDisplay.hpp

```
#ifndef __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
#define __STDAIR_BAS_BASCONST_BOMMANAGER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost
#include <boost/tokenizer.hpp>

namespace stdair {

  extern const std::string DISPLAY_LEVEL_STRING_ARRAY[51];

  extern const std::string DEFAULT_KEY_FLD_DELIMITER;

  extern const std::string DEFAULT_KEY_SUB_FLD_DELIMITER;

  extern const boost::char_separator<char> DEFAULT_KEY_TOKEN_DELIMITER;

}
#endif // __STDAIR_BAS_BASCONST_BOMMANAGER_HPP
```

33.25 stdair/basic/BasConst_BookingClass.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_fare_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const Distance_T stdair::DEFAULT_DISTANCE_VALUE
- const ClassCode_T stdair::DEFAULT_CLOSED_CLASS_CODE
- const NbOfBookings_T stdair::DEFAULT_CLASS_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS
- const NbOfBookings_T stdair::DEFAULT_CLASS_UNCONSTRAINED_DEMAND
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_MEAN
- const NbOfBookings_T stdair::DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_-DEVIATION
- const NbOfCancellations_T stdair::DEFAULT_CLASS_NB_OF_CANCELLATIONS
- const NbOfNoShows_T stdair::DEFAULT_CLASS_NB_OF_NOSHOWS
- const CabinCapacity_T stdair::DEFAULT_CABIN_CAPACITY
- const CommittedSpace_T stdair::DEFAULT_COMMITTED_SPACE
- const BlockSpace_T stdair::DEFAULT_BLOCK_SPACE
- const Availability_T stdair::DEFAULT_NULL_AVAILABILITY
- const Availability_T stdair::DEFAULT_AVAILABILITY
- const CensorshipFlag_T stdair::DEFAULT_CLASS_CENSORSHIPFLAG
- const BookingLimit_T stdair::DEFAULT_CLASS_BOOKING_LIMIT
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL
- const AuthorizationLevel_T stdair::DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL
- const OverbookingRate_T stdair::DEFAULT_CLASS_OVERBOOKING_RATE
- const Fare_T stdair::DEFAULT_FARE_VALUE
- const Revenue_T stdair::DEFAULT_REVENUE_VALUE
- const PriceCurrency_T stdair::DEFAULT_CURRENCY
- const Percentage_T stdair::DEFAULT_LOAD_FACTOR_VALUE
- const DayDuration_T stdair::DEFAULT_DAY_DURATION
- const double stdair::DEFAULT_EPSILON_VALUE

33.26 stdair/basic/BasConst_BookingClass.hpp

```
#ifndef __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
#define __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_fare_types.hpp>

namespace stdair {

  // //////// (Segment-)Class-related BOM ///////
  extern const Distance_T DEFAULT_DISTANCE_VALUE;

  extern const ClassCode_T DEFAULT_CLOSED_CLASS_CODE;

  extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;

  extern const NbOfBookings_T DEFAULT_CLASS_TOTAL_NB_OF_BOOKINGS;

  extern const NbOfBookings_T DEFAULT_CLASS_UNCONSTRAINED_DEMAND;

  extern const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_MEAN;

  extern const NbOfBookings_T DEFAULT_CLASS_REMAINING_DEMAND_STANDARD_DEVIATION;

  extern const NbOfCancellations_T DEFAULT_CLASS_NB_OF_CANCELLATIONS;

  extern const NbOfNoShows_T DEFAULT_CLASS_NB_OF_NOSHOWS;

  extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;

  extern const CommittedSpace_T DEFAULT_COMMITTED_SPACE;

  extern const BlockSpace_T DEFAULT_BLOCK_SPACE;

  extern const Availability_T DEFAULT_NULL_AVAILABILITY;

  extern const Availability_T DEFAULT_AVAILABILITY;

  extern const CensorshipFlag_T DEFAULT_CLASS_CENSORSHIPFLAG;

  extern const CensorshipFlagList_T DEFAULT_CLASS_CENSORSHIPFLAG_LIST;

  extern const BookingLimit_T DEFAULT_CLASS_BOOKING_LIMIT;

  extern const AuthorizationLevel_T DEFAULT_CLASS_AUTHORIZATION_LEVEL;

  extern const AuthorizationLevel_T DEFAULT_CLASS_MAX_AUTHORIZATION_LEVEL;

  extern const AuthorizationLevel_T DEFAULT_CLASS_MIN_AUTHORIZATION_LEVEL;

  extern const OverbookingRate_T DEFAULT_CLASS_OVERBOOKING_RATE;

  extern const Fare_T DEFAULT_FARE_VALUE;

  extern const Revenue_T DEFAULT_REVENUE_VALUE;

  extern const PriceCurrency_T DEFAULT_CURRENCY;

  extern const Percentage_T DEFAULT_LOAD_FACTOR_VALUE;

```

```
00097   extern const DayDuration_T DEFAULT_DAY_DURATION;
00098
00101   extern const double DEFAULT_EPSILON_VALUE;
00102
00103 }
00104 #endif // __STDAIR_BAS_BASCONST_BOOKINGCLASS_HPP
```

33.27 stdair/basic/BasConst_DefaultObject.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const AirportCode_T stdair::AIRPORT_LHR
- const AirportCode_T stdair::AIRPORT_SYD
- const CityCode_T stdair::POS_LHR
- const DayDuration_T stdair::NO_ADVANCE_PURCHASE
- const SaturdayStay_T stdair::SATURDAY_STAY
- const SaturdayStay_T stdair::NO_SATURDAY_STAY
- const ChangeFees_T stdair::CHANGE_FEES
- const ChangeFees_T stdair::NO_CHANGE_FEES
- const NonRefundable_T stdair::NON_REFUNDABLE
- const NonRefundable_T stdair::NO_NON_REFUNDABLE
- const DayDuration_T stdair::NO_STAY_DURATION
- const CabinCode_T stdair::CABIN_Y
- const AirlineCode_T stdair::AIRLINE_CODE_BA
- const ClassCode_T stdair::CLASS_CODE_Y
- const ClassCode_T stdair::CLASS_CODE_Q
- const AirportCode_T stdair::AIRPORT_SIN
- const AirportCode_T stdair::AIRPORT_BKK
- const CityCode_T stdair::POS_SIN
- const CabinCode_T stdair::CABIN_ECO
- const FrequentFlyer_T stdair::FREQUENT_FLYER_MEMBER

33.28 stdair/basic/BasConst_DefaultObject.hpp

```
#ifndef __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
#define __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_types.hpp>

namespace stdair {

  // //////// Fare and Yield related BOM Tree//////
  extern const AirportCode_T AIRPORT_LHR;

  extern const AirportCode_T AIRPORT_SYD;

  extern const CityCode_T POS_LHR;

  extern const DayDuration_T NO_ADVANCE_PURCHASE;

  extern const SaturdayStay_T SATURDAY_STAY;

  extern const SaturdayStay_T NO_SATURDAY_STAY;

  extern const ChangeFees_T CHANGE_FEES;

  extern const ChangeFees_T NO_CHANGE_FEES;

  extern const NonRefundable_T NON_REFUNDABLE;

  extern const NonRefundable_T NO_NON_REFUNDABLE;

  extern const DayDuration_T NO_STAY_DURATION;

  extern const CabinCode_T CABIN_Y;

  extern const AirlineCode_T AIRLINE_CODE_BA;

  extern const ClassCode_T CLASS_CODE_Y;

  // //////// Travel Solution related objects//////
  extern const ClassCode_T CLASS_CODE_Q;

  // //////// Booking request related objects//////
  extern const AirportCode_T AIRPORT_SIN;

  extern const AirportCode_T AIRPORT_BKK;

  extern const CityCode_T POS_SIN;

  extern const CabinCode_T CABIN_ECO;

  extern const FrequentFlyer_T FREQUENT_FLYER_MEMBER;

}
#endif // __STDAIR_BAS_BASCONST_DEFAULTOBJECT_HPP
```

33.29 stdair/basic/BasConst_Event.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const Count_T stdair::DEFAULT_PROGRESS_STATUS
- const Date_T stdair::DEFAULT_EVENT_OLDEST_DATE
- const DateTime_T stdair::DEFAULT_EVENT_OLDEST_DATETIME
- const Percentage_T stdair::MAXIMUM_PROGRESS_STATUS

33.30 stdair/basic/BasConst_Event.hpp

```
#ifndef __STDAIR_BAS_BASCONST_EVENT_HPP
#define __STDAIR_BAS_BASCONST_EVENT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>

namespace stdair {

  extern const Count_T DEFAULT_PROGRESS_STATUS;

  extern const Date_T DEFAULT_EVENT_OLDEST_DATE;

  extern const DateTime_T DEFAULT_EVENT_OLDEST_DATETIME;

  extern const Percentage_T MAXIMUM_PROGRESS_STATUS;

}
#endif // __STDAIR_BAS_BASCONST_EVENT_HPP
```

33.31 stdair/basic/BasConst_General.hpp File Reference

```
#include <string>
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const std::string stdair::DEFAULT_BOM_ROOT_KEY
- const NbOfFlightDates_T stdair::DEFAULT_NB_OF_FLIGHTDATES
- const unsigned int stdair::DEFAULT_FLIGHT_SPEED
- const BookingRatio_T stdair::DEFAULT_OND_BOOKING_RATE
- const Count_T stdair::SECONDS_IN_ONE_DAY
- const Count_T stdair::MILLISECONDS_IN_ONE_SECOND
- const Date_T stdair::DEFAULT_DATE
- const DateTime_T stdair::DEFAULT_DATETIME
- const Duration_T stdair::DEFAULT_EPSILON_DURATION
- const RandomSeed_T stdair::DEFAULT_RANDOM_SEED
- const Duration_T stdair::NULL_BOOST_TIME_DURATION
- const Duration_T stdair::DEFAULT_NULL_DURATION
- const Fare_T stdair::DEFAULT_CLASS_FARE_VALUE
- const NbOfAirlines_T stdair::DEFAULT_NBOFAIRLINES
- const unsigned int stdair::DEFAULT_NB_OF_DAYS_IN_A_YEAR
- const ChannelLabel_T stdair::DEFAULT_CHANNEL
- const unsigned int stdair::DEFAULT_NUMBER_OF_SUBDIVISIONS

33.32 stdair/basic/BasConst_General.hpp

```
#ifndef __STDAIR_BAS_BASCONST_GENERAL_HPP
#define __STDAIR_BAS_BASCONST_GENERAL_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_types.hpp>

namespace stdair {

  extern const std::string DEFAULT_BOM_ROOT_KEY;

  extern const double DEFAULT_EPSILON_VALUE;

  extern const CabinCapacity_T DEFAULT_CABIN_CAPACITY;

  extern const NbOfFlightDates_T DEFAULT_NB_OF_FLIGHTDATES;

  extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;

  extern const Distance_T DEFAULT_DISTANCE_VALUE;

  extern const unsigned int DEFAULT_FLIGHT_SPEED;

  extern const Fare_T DEFAULT_FARE_VALUE;

  extern const PriceCurrency_T DEFAULT_CURRENCY;

  extern const Revenue_T DEFAULT_REVENUE_VALUE;

  extern const BookingRatio_T DEFAULT_OND_BOOKING_RATE;

  extern const Count_T SECONDS_IN_ONE_DAY;

  extern const Count_T MILLISECONDS_IN_ONE_SECOND;

  extern const Date_T DEFAULT_DATE;

  extern const DateTime_T DEFAULT_DATETIME;

  extern const Duration_T DEFAULT_EPSILON_DURATION;

  extern const RandomSeed_T DEFAULT_RANDOM_SEED;

  extern const Duration_T NULL_BOOST_TIME_DURATION;

  extern const Duration_T DEFAULT_NULL_DURATION;

  extern const Fare_T DEFAULT_CLASS_FARE_VALUE;

  extern const NbOfAirlines_T DEFAULT_NBOFAIRLINES;

  extern const unsigned int DEFAULT_NB_OF_DAYS_IN_A_YEAR;

  extern const NbOfBookings_T DEFAULT_CLASS_NB_OF_BOOKINGS;

  extern const ChannelLabel_T DEFAULT_CHANNEL;

  extern const OnDStringList_T DEFAULT_OND_STRING_LIST;

  extern const unsigned int DEFAULT_NUMBER_OF_SUBDIVISIONS;

```

```
00092 }
00093 #endif // __STDAIR_BAS_BASCONST_GENERAL_HPP
```

33.33 stdair/basic/BasConst_Inventory.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
```

Classes

- struct stdair::DefaultDCPList
- struct stdair::DefaultDtdFratMap
- struct stdair::DefaultDtdProbMap

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const AirlineCode_T stdair::DEFAULT_AIRLINE_CODE
- const AirlineCode_T stdair::DEFAULT_NULL_AIRLINE_CODE
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER
- const FlightNumber_T stdair::DEFAULT_FLIGHT_NUMBER_FF
- const TableID_T stdair::DEFAULT_TABLE_ID
- const Date_T stdair::DEFAULT_DEPARTURE_DATE
- const AirportCode_T stdair::DEFAULT_AIRPORT_CODE
- const AirportCode_T stdair::DEFAULT_NULL_AIRPORT_CODE
- const AirportCode_T stdair::DEFAULT_ORIGIN
- const AirportCode_T stdair::DEFAULT_DESTINATION
- const CabinCode_T stdair::DEFAULT_CABIN_CODE
- const FamilyCode_T stdair::DEFAULT_FARE_FAMILY_CODE
- const FamilyCode_T stdair::DEFAULT_NULL_FARE_FAMILY_CODE
- const PolicyCode_T stdair::DEFAULT_POLICY_CODE
- const NestingStructureCode_T stdair::DEFAULT_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T stdair::DISPLAY_NESTING_STRUCTURE_CODE
- const NestingStructureCode_T stdair::YIELD_BASED_NESTING_STRUCTURE_CODE
- const NestingNodeCode_T stdair::DEFAULT_NESTING_NODE_CODE
- const ClassCode_T stdair::DEFAULT_CLASS_CODE
- const ClassCode_T stdair::DEFAULT_NULL_CLASS_CODE
- const BidPrice_T stdair::DEFAULT_BID_PRICE
- const unsigned short stdair::MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT
- const unsigned short stdair::MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND

- const Availability_T stdair::MAXIMAL_AVAILABILITY
- const SeatIndex_T stdair::DEFAULT_SEAT_INDEX
- const NbOfSeats_T stdair::DEFAULT_NULL_BOOKING_NUMBER
- const CapacityAdjustment_T stdair::DEFAULT_NULL_CAPACITY_ADJUSTMENT
- const UPR_T stdair::DEFAULT_NULL_UPR
- const std::string stdair::DEFAULT_FARE_FAMILY_VALUE_TYPE
- const std::string stdair::DEFAULT_SEGMENT_CABIN_VALUE_TYPE

33.34 stdair/basic/BasConst_Inventory.hpp

```
#ifndef __STDAIR_BAS_BASCONST_INVENTORY_HPP
#define __STDAIR_BAS_BASCONST_INVENTORY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>

namespace stdair {

  // //////// Inventory-related BOM ///////
  extern const AirlineCode_T DEFAULT_AIRLINE_CODE;

  extern const AirlineCode_T DEFAULT_NULL_AIRLINE_CODE;

  extern const AirlineCodeList_T DEFAULT_AIRLINE_CODE_LIST;

  extern const FlightNumber_T DEFAULT_FLIGHT_NUMBER;

  extern const FlightNumber_T DEFAULT_FLIGHT_NUMBER_FF;

  extern const TableID_T DEFAULT_TABLE_ID;

  extern const Date_T DEFAULT_DEPARTURE_DATE;

  extern const AirportCode_T DEFAULT_AIRPORT_CODE;

  extern const AirportCode_T DEFAULT_NULL_AIRPORT_CODE;

  extern const AirportCode_T DEFAULT_ORIGIN;

  extern const AirportCode_T DEFAULT_DESTINATION;

  extern const CabinCode_T DEFAULT_CABIN_CODE;

  extern const FamilyCode_T DEFAULT_FARE_FAMILY_CODE;

  extern const FamilyCode_T DEFAULT_NULL_FARE_FAMILY_CODE;

  extern const PolicyCode_T DEFAULT_POLICY_CODE;

  extern const NestingStructureCode_T DEFAULT_NESTING_STRUCTURE_CODE;

  extern const NestingStructureCode_T DISPLAY_NESTING_STRUCTURE_CODE;

  extern const NestingStructureCode_T YIELD_BASED_NESTING_STRUCTURE_CODE;

  extern const NestingNodeCode_T DEFAULT_NESTING_NODE_CODE;

  extern const ClassCode_T DEFAULT_CLASS_CODE;

  extern const ClassCode_T DEFAULT_NULL_CLASS_CODE;

  extern const ClassList_StringList_T DEFAULT_CLASS_CODE_LIST;

  extern const BidPrice_T DEFAULT_BID_PRICE;

  extern const BidPriceVector_T DEFAULT_BID_PRICE_VECTOR;
```

```
00090
00094   extern const unsigned short MAXIMAL_NUMBER_OF_LEGS_IN_FLIGHT;
00095
00098   extern const unsigned short MAXIMAL_NUMBER_OF_SEGMENTS_IN_OND;
00099
00101   extern const Availability_T MAXIMAL_AVAILABILITY;
00102
00104   extern const SeatIndex_T DEFAULT_SEAT_INDEX;
00105
00107   extern const NbOfSeats_T DEFAULT_NULL_BOOKING_NUMBER;
00108
00110   extern const CapacityAdjustment_T DEFAULT_NULL_CAPACITY_ADJUSTMENT;
00111
00113   extern const UPR_T DEFAULT_NULL_UPR;
00114
00116   extern const std::string DEFAULT_FARE_FAMILY_VALUE_TYPE;
00117
00119   extern const std::string DEFAULT_SEGMENT_CABIN_VALUE_TYPE;
00120
00122   extern const int DEFAULT_MAX_DTD;
00123
00125   extern const DCPList_T DEFAULT_DCP_LIST;
00126   struct DefaultDCPList { static DCPList_T init(); };
00127
00129   extern const DTDFratMap_T DEFAULT_DTD_FRAT5COEF_MAP;
00130   struct DefaultDtdFratMap { static DTDFratMap_T init();};
00131
00133   extern const DTDProbMap_T DEFAULT_DTD_PROB_MAP;
00134   struct DefaultDtdProbMap { static DTDProbMap_T init();};
00135
00137   extern const ForecastingMethod DEFAULT_FORECASTING_METHOD;
00138
00140   extern const UnconstrainingMethod DEFAULT_UNCONSTRAINING_METHOD;
00141
00143   extern const PreOptimisationMethod DEFAULT_PREOPTIMISATION_METHOD;
00144
00146   extern const OptimisationMethod DEFAULT_OPTIMISATION_METHOD;
00147
00149   extern const PartnershipTechnique DEFAULT_PARTNERSHIP_TECHNIQUE;
00150
00151 }
00152 #endif // __STDAIR_BAS_BASCONST_INVENTORY_HPP
```

33.35 stdair/basic/BasConst_Period_BOM.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const DatePeriod_T stdair::BOOST_DEFAULT_DATE_PERIOD
- const DOW_String_T stdair::DEFAULT_DOW_STRING
- const DateOffset_T stdair::DEFAULT_DATE_OFFSET

33.36 stdair/basic/BasConst_Period_BOM.hpp

```
#ifndef __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
#define __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_types.hpp>

namespace stdair {

  // //////// (Flight-)Period-related BOM ///////
  extern const DatePeriod_T BOOST_DEFAULT_DATE_PERIOD;

  extern const std::string DOW_STR[];

  extern const DOW_String_T DEFAULT_DOW_STRING;

  extern const DateOffset_T DEFAULT_DATE_OFFSET;

  extern const DayDuration_T DEFAULT_DAY_DURATION;

}
#endif // __STDAIR_BAS_BASCONST_PERIOD_BOM_HPP
```

33.37 stdair/basic/BasConst_Request.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const PartySize_T stdair::DEFAULT_PARTY_SIZE
- const DayDuration_T stdair::DEFAULT_STAY_DURATION
- const WTP_T stdair::DEFAULT_WTP
- const CityCode_T stdair::DEFAULT_POS
- const Date_T stdair::DEFAULT_PREFERRED_DEPARTURE_DATE
- const Duration_T stdair::DEFAULT_PREFERRED_DEPARTURE_TIME
- const DateOffset_T stdair::DEFAULT_ADVANCE_PURCHASE
- const Date_T stdair::DEFAULT_REQUEST_DATE
- const Duration_T stdair::DEFAULT_REQUEST_TIME
- const DateTime_T stdair::DEFAULT_REQUEST_DATE_TIME
- const CabinCode_T stdair::DEFAULT_PREFERRED_CABIN
- const ChannelLabel_T stdair::CHANNEL_DN
- const ChannelLabel_T stdair::CHANNEL_IN
- const TripType_T stdair::TRIP_TYPE_ONE_WAY
- const TripType_T stdair::TRIP_TYPE_ROUND_TRIP
- const TripType_T stdair::TRIP_TYPE_INBOUND
- const TripType_T stdair::TRIP_TYPE_OUTBOUND
- const FrequentFlyer_T stdair::DEFAULT_FF_TIER
- const PriceValue_T stdair::DEFAULT_VALUE_OF_TIME
- const IntDuration_T stdair::HOUR_CONVERTED_IN_SECONDS

33.38 stdair/basic/BasConst_Request.hpp

```
#ifndef __STDAIR_BAS_BASCONST_REQUEST_HPP
#define __STDAIR_BAS_BASCONST_REQUEST_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

namespace stdair {

  extern const PartySize_T DEFAULT_PARTY_SIZE;

  extern const DayDuration_T DEFAULT_STAY_DURATION;

  extern const WTP_T DEFAULT_WTP;

  extern const CityCode_T DEFAULT_POS;

  extern const Date_T DEFAULT_PREFERRED_DEPARTURE_DATE;

  extern const Duration_T DEFAULT_PREFERRED_DEPARTURE_TIME;

  extern const DateOffset_T DEFAULT_ADVANCE_PURCHASE;

  extern const Date_T DEFAULT_REQUEST_DATE;

  extern const Duration_T DEFAULT_REQUEST_TIME;

  extern const DateTime_T DEFAULT_REQUEST_DATE_TIME;

  extern const CabinCode_T DEFAULT_PREFERRED_CABIN;

  extern const ChannelLabel_T DEFAULT_CHANNEL;

  extern const ChannelLabel_T CHANNEL_DN;

  extern const ChannelLabel_T CHANNEL_IN;

  extern const TripType_T TRIP_TYPE_ONE_WAY;

  extern const TripType_T TRIP_TYPE_ROUND_TRIP;

  extern const TripType_T TRIP_TYPE_INBOUND;

  extern const TripType_T TRIP_TYPE_OUTBOUND;

  extern const FrequentFlyer_T DEFAULT_FF_TIER;

  extern const PriceValue_T DEFAULT_VALUE_OF_TIME;

  extern const IntDuration_T HOUR_CONVERTED_IN_SECONDS;

}
#endif // __STDAIR_BAS_BASCONST_REQUEST_HPP
```

33.39 stdair/basic/BasConst_SellUpCurves.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Classes

- struct stdair::DefaultMap

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.40 stdair/basic/BasConst_SellUpCurves.hpp

```
#ifndef __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP
#define __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/stdair_types.hpp>

namespace stdair {

  extern const FRAT5Curve_T FRAT5_CURVE_A;
  extern const FRAT5Curve_T FRAT5_CURVE_B;
  extern const FRAT5Curve_T FRAT5_CURVE_C;
  extern const FRAT5Curve_T FRAT5_CURVE_D;

  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_A;
  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_B;
  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_C;
  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_D;
  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_E;
  extern const FFDisutilityCurve_T FF_DISUTILITY_CURVE_F;

  struct DefaultMap {
    static FRAT5Curve_T createFRAT5CurveA();
    static FRAT5Curve_T createFRAT5CurveB();
    static FRAT5Curve_T createFRAT5CurveC();
    static FRAT5Curve_T createFRAT5CurveD();
    static FFDisutilityCurve_T createFFDisutilityCurveA();
    static FFDisutilityCurve_T createFFDisutilityCurveB();
    static FFDisutilityCurve_T createFFDisutilityCurveC();
    static FFDisutilityCurve_T createFFDisutilityCurveD();
    static FFDisutilityCurve_T createFFDisutilityCurveE();
    static FFDisutilityCurve_T createFFDisutilityCurveF();
  };
}
#endif // __STDAIR_BAS_BASCONST_SELLUPCURVES_HPP
```

33.41 stdair/basic/BasConst_TravelSolution.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const Duration_T stdair::DEFAULT_MINIMAL_CONNECTION_TIME
- const Duration_T stdair::DEFAULT_MAXIMAL_CONNECTION_TIME
- const FlightPathCode_T stdair::DEFAULT_FLIGHTPATH_CODE
- const Availability_T stdair::DEFAULT_CLASS_AVAILABILITY
- const AvailabilityStatus_T stdair::DEFAULT_AVAILABILITY_STATUS
- const unsigned short stdair::DEFAULT_NUMBER_OF_REQUIRED_SEATS
- const MatchingIndicator_T stdair::DEFAULT_MATCHING_INDICATOR
- const AirlineCode_T stdair::DEFAULT_DICO_STUDIED_AIRLINE

33.42 stdair/basic/BasConst_TravelSolution.hpp

```
#ifndef __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
#define __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_types.hpp>

namespace stdair {

  // //////// Travel Solutions ///////
  extern const Distance_T DEFAULT_DISTANCE_VALUE;

  extern const Duration_T DEFAULT_MINIMAL_CONNECTION_TIME;

  extern const Duration_T DEFAULT_MAXIMAL_CONNECTION_TIME;

  extern const Duration_T NULL_BOOST_TIME_DURATION;

  extern const FlightPathCode_T DEFAULT_FLIGHTPATH_CODE;

  extern const Availability_T DEFAULT_CLASS_AVAILABILITY;

  extern const AvailabilityStatus_T DEFAULT_AVAILABILITY_STATUS;

  extern const unsigned short DEFAULT_NUMBER_OF_REQUIRED_SEATS;

  extern const MatchingIndicator_T DEFAULT_MATCHING_INDICATOR;

  extern const Revenue_T DEFAULT_REVENUE_VALUE;

  extern const AirlineCode_T DEFAULT_DICO_STUDIED_AIRLINE;

  extern const Date_T DEFAULT_DICO_STUDIED_DATE;

}
#endif // __STDAIR_BAS_BASCONST_TRAVELSOLUTION_HPP
```

33.43 stdair/basic/BasConst_Yield.hpp File Reference

```
#include <stdair/stdair_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Variables

- const Yield_T stdair::DEFAULT_YIELD_VALUE
- const Yield_T stdair::DEFAULT_YIELD_MAX_VALUE

33.44 stdair/basic/BasConst_Yield.hpp

```
#ifndef __STDAIR_BAS_BASCONST_YIELD_HPP
#define __STDAIR_BAS_BASCONST_YIELD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_types.hpp>

namespace stdair {

  // //////// (Leg-)Yield-related BOM ///////
  extern const Yield_T DEFAULT_YIELD_VALUE;

  extern const Yield_T DEFAULT_YIELD_MAX_VALUE;

}
#endif // __STDAIR_BAS_BASCONST_YIELD_HPP
```

33.45 stdair/basic/BasDBParams.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasDBParams.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.46 stdair/basic/BasDBParams.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasDBParams.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BasDBParams::BasDBParams() {
  }

  // ////////////////////////////////////////////////////////////////////
  BasDBParams::BasDBParams (const BasDBParams& iDBParams)
    : _user (iDBParams._user), _passwd (iDBParams._passwd),
      _host (iDBParams._host), _port (iDBParams._port),
      _dbname (iDBParams._dbname) {
  }

  // ////////////////////////////////////////////////////////////////////
  BasDBParams::BasDBParams (const std::string& iDBUser,
                            const std::string& iDBPasswd,
                            const std::string& iDBHost,
                            const std::string& iDBPort,
                            const std::string& iDBName)
    : _user (iDBUser), _passwd (iDBPasswd), _host (iDBHost), _port (iDBPort),
      _dbname (iDBName) {
  }

  // ////////////////////////////////////////////////////////////////////
  BasDBParams::~BasDBParams() {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BasDBParams::describe() const {
    return toString();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BasDBParams::toShortString() const {
    std::ostringstream oStr;
    oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BasDBParams::toString() const {
    std::ostringstream oStr;
    oStr << _dbname << "." << _user << "@" << _host << ":" << _port;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool BasDBParams::check() const {
    if (_user.empty() == true || _passwd.empty() == true
        || _host.empty() == true || _port.empty()
        || _dbname.empty() == true) {
      return false;
    }
    return true;
  }

```

```
00066 }
```

33.47 stdair/basic/BasDBParams.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_db.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::BasDBParams

  Structure holding the parameters for connection to a database.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.48 stdair/basic/BasDBParams.hpp

```
#ifndef __STDAIR_BAS_BASDBPARAMS_HPP
#define __STDAIR_BAS_BASDBPARAMS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// Stdair
#include <stdair/stdair_db.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct BasDBParams : public StructAbstract {
  public:
    // ///////// Getters ////////
    const std::string& getUser() const {
      return _user;
    }

    const std::string& getPassword() const {
      return _passwd;
    }

    const std::string& getHost() const {
      return _host;
    }

    const std::string& getPort() const {
      return _port;
    }

    const std::string& getDBName() const {
      return _dbname;
    }


    // ///////// Setters //////////
    void setUser (const std::string& iUser) {
      _user = iUser;
    }

    void setPassword (const std::string& iPasswd) {
      _passwd = iPasswd;
    }

    void setHost (const std::string& iHost) {
      _host = iHost;
    }

    void setPort (const std::string& iPort) {
      _port = iPort;
    }

    void setDBName (const std::string& iDBName) {
      _dbname = iDBName;
    }


  public:
    // ///////// Busines methods ////////
    bool check() const;

```

```

  public:
    // ///////// Display methods ////////
    const std::string describe() const;

    std::string toShortString() const;

    std::string toString() const;


  public:
    BasDBParams (const std::string& iDBUser, const std::string& iDBPasswd,
                 const std::string& iDBHost, const std::string& iDBPort,
                 const std::string& iDBName);

    BasDBParams();

    BasDBParams (const BasDBParams&);

    ~BasDBParams();


  private:
    // /////// Attributes /////////
    std::string _user;
    std::string _passwd;
    std::string _host;
    std::string _port;
    std::string _dbname;
  };

}
#endif // __STDAIR_BAS_BASDBPARAMS_HPP
```

33.49 stdair/basic/BasFileMgr.cpp File Reference

```
#include <cassert>
#include <boost/version.hpp>
#include <boost/filesystem/path.hpp>
#include <boost/filesystem/operations.hpp>
#include <stdair/basic/BasFileMgr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.50 stdair/basic/BasFileMgr.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// Boost (STL Extension)
// Boost Filesystem (http://www.boost.org/doc/libs/1_41_0/libs/filesystem/doc/index.htm)
#include <boost/version.hpp>
#if BOOST_VERSION >= 103500
#include <boost/filesystem.hpp>
#else // BOOST_VERSION >= 103500
#include <boost/filesystem/path.hpp>
#include <boost/filesystem/operations.hpp>
#endif // BOOST_VERSION >= 103500
// StdAir
#include <stdair/basic/BasFileMgr.hpp>

namespace boostfs = boost::filesystem;

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  bool BasFileMgr::doesExistAndIsReadable (const std::string& iFilepath) {
    bool oFine = false;

    boostfs::path lPath (iFilepath);

    if (boostfs::exists (lPath) == false) {
      return oFine;
    }

#if BOOST_VERSION >= 103500
    if (boostfs::is_regular (lPath) == true) {
      oFine = true;
    }
#endif // BOOST_VERSION >= 103500

    return oFine;
  }

}
```

33.51 stdair/basic/BasFileMgr.hpp File Reference

```
#include <string>
```

Classes

- struct stdair::BasFileMgr

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.52 stdair/basic/BasFileMgr.hpp

```
#ifndef __STDAIR_BAS_BASFILEMGR_HPP
#define __STDAIR_BAS_BASFILEMGR_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

  struct BasFileMgr {
  public:

    // ///////////// Functional Support Methods ////////////
    static bool doesExistAndIsReadable (const std::string& iFilepath);

  };

}
#endif // __STDAIR_BAS_BASFILEMGR_HPP
```

33.53 stdair/basic/BasLogParams.cpp File Reference

```
#include <cassert>
#include <iostream>
#include <sstream>
#include <stdair/basic/BasLogParams.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.54 stdair/basic/BasLogParams.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <iostream>
#include <sstream>
// StdAir
#include <stdair/basic/BasLogParams.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BasLogParams::BasLogParams()
    : _logLevel (LOG::DEBUG), _logStream (std::cout),
      _forceMultipleInit (false) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  BasLogParams::BasLogParams (const BasLogParams& iLogParams)
    : _logLevel (iLogParams._logLevel), _logStream (iLogParams._logStream),
      _forceMultipleInit (iLogParams._forceMultipleInit) {
  }

  // ////////////////////////////////////////////////////////////////////
  BasLogParams::BasLogParams (const LOG::EN_LogLevel iLogLevel,
                              std::ostream& ioLogOutputStream,
                              const bool iForceMultipleInstance)
    : _logLevel (iLogLevel), _logStream (ioLogOutputStream),
      _forceMultipleInit (iForceMultipleInstance) {
  }

  // ////////////////////////////////////////////////////////////////////
  BasLogParams::~BasLogParams() {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BasLogParams::describe() const {
    return toString();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BasLogParams::toShortString() const {
    const std::string isForcedStr = (_forceMultipleInit == true)?" (forced)":"";
    std::ostringstream oStr;
    oStr << LOG::_logLevels[_logLevel] << isForcedStr;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BasLogParams::toString() const {
    std::ostringstream oStr;
    oStr << LOG::_logLevels[_logLevel];
    return oStr.str();
  }

}
```

33.55 stdair/basic/BasLogParams.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::BasLogParams

  Structure holding parameters for logging.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.56 stdair/basic/BasLogParams.hpp

```
#ifndef __STDAIR_BAS_BASLOGPARAMS_HPP
#define __STDAIR_BAS_BASLOGPARAMS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// Stdair
#include <stdair/stdair_log.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct BasLogParams : public StructAbstract {
    friend class Logger;
  public:
    // //////// Getters ////////
    const LOG::EN_LogLevel& getLogLevel() const {
      return _logLevel;
    }

    std::ostream& getLogStream() const {
      return _logStream;
    }

    const bool getForcedInitialisationFlag() const {
      return _forceMultipleInit;
    }


    // //////// Setters //////////
    void setForcedInitialisationFlag (const bool iForceMultipleInstance) {
      _forceMultipleInit = iForceMultipleInstance;
    }


  public:
    // //////// Busines methods ////////
    bool check() const;


  public:
    // //////// Display methods ////////
    const std::string describe() const;

    std::string toShortString() const;

    std::string toString() const;


  public:
    BasLogParams (const LOG::EN_LogLevel iLogLevel,
                  std::ostream& ioLogOutputStream,
                  const bool iForceMultipleInstance = false);

    BasLogParams (const BasLogParams&);

    ~BasLogParams();

  private:
    BasLogParams();


```

```
  private:
    // /////// Attributes /////////
    const LOG::EN_LogLevel _logLevel;

    std::ostream& _logStream;

    bool _forceMultipleInit;
  };

}
#endif // __STDAIR_BAS_BASLOGPARAMS_HPP
```

33.57 stdair/basic/BasParserHelperTypes.hpp File Reference

```
#include <string>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>
```

Classes

- struct stdair::date_time_element< MIN, MAX >

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef date_time_element< 0, 23 > stdair::hour_t
- typedef date_time_element< 0, 59 > stdair::minute_t
- typedef date_time_element< 0, 59 > stdair::second_t
- typedef date_time_element< 1900, 2100 > stdair::year_t
- typedef date_time_element< 1, 12 > stdair::month_t
- typedef date_time_element< 1, 31 > stdair::day_t

Functions

- template<int MIN, int MAX>
  date_time_element< MIN, MAX > stdair::operator* (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)
- template<int MIN, int MAX>
  date_time_element< MIN, MAX > stdair::operator+ (const date_time_element< MIN, MAX > &o1, const date_time_element< MIN, MAX > &o2)

33.58 stdair/basic/BasParserHelperTypes.hpp

```
#ifndef __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
#define __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  //
  // Parser structure helper
  //
  // ////////////////////////////////////////////////////////////////////
  template <int MIN = 0, int MAX = 0>
  struct date_time_element {
    unsigned int _value;

    // ////////// Constructors ///////////
    date_time_element () { }
    date_time_element (const date_time_element& t) : _value (t._value) { }
    date_time_element (int i) : _value (i) { }
    void check () const {
      if (_value < MIN || _value > MAX) {
        std::ostringstream oMessage;
        oMessage << "The value: " << _value << " is out of range ("
                 << MIN << ", " << MAX << ")";
        throw stdair::ParserException (oMessage.str());
      }
    }
  };

  template <int MIN, int MAX>
  inline date_time_element<MIN,
                           MAX> operator*(const date_time_element<MIN, MAX>& o1,
                                          const date_time_element<MIN, MAX>& o2){

    return date_time_element<MIN, MAX> (o1._value * o2._value);
  }

  template <int MIN, int MAX>
  inline date_time_element<MIN,
                           MAX> operator+(const date_time_element<MIN, MAX>& o1,
                                          const date_time_element<MIN, MAX>& o2){

    return date_time_element<MIN, MAX> (o1._value + o2._value);
  }

  typedef date_time_element<0, 23> hour_t;
  typedef date_time_element<0, 59> minute_t;
  typedef date_time_element<0, 59> second_t;
  typedef date_time_element<1900, 2100> year_t;
  typedef date_time_element<1, 12> month_t;
  typedef date_time_element<1, 31> day_t;

}
#endif // __STDAIR_BAS_BASCOMPARSERHELPERTYPES_HPP
```

33.59 stdair/basic/BasParserTypes.hpp File Reference

```
#include <string>
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/spirit/include/support_multi_pass.hpp>
#include <stdair/basic/BasParserHelperTypes.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::istreambuf_iterator< char > stdair::base_iterator_t
- typedef boost::spirit::multi_pass< base_iterator_t > stdair::iterator_t
- typedef boost::spirit::qi::int_parser< unsigned int, 10, 1, 1 > stdair::int1_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 2, 2 > stdair::uint2_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 4, 4 > stdair::uint4_p_t
- typedef boost::spirit::qi::uint_parser< int, 10, 1, 4 > stdair::uint1_4_p_t
- typedef boost::spirit::qi::uint_parser< hour_t, 10, 2, 2 > stdair::hour_p_t
- typedef boost::spirit::qi::uint_parser< minute_t, 10, 2, 2 > stdair::minute_p_t
- typedef boost::spirit::qi::uint_parser< second_t, 10, 2, 2 > stdair::second_p_t
- typedef boost::spirit::qi::uint_parser< year_t, 10, 4, 4 > stdair::year_p_t
- typedef boost::spirit::qi::uint_parser< month_t, 10, 2, 2 > stdair::month_p_t
- typedef boost::spirit::qi::uint_parser< day_t, 10, 2, 2 > stdair::day_p_t

33.60 stdair/basic/BasParserTypes.hpp

```
#ifndef __STDAIR_BAS_BASCOMPARSERTYPES_HPP
#define __STDAIR_BAS_BASCOMPARSERTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost Spirit (Parsing)
#include <boost/spirit/include/qi.hpp>
#include <boost/spirit/include/phoenix_core.hpp>
#include <boost/spirit/include/phoenix_operator.hpp>
#include <boost/spirit/include/support_multi_pass.hpp>
// STDAIR
#include <stdair/basic/BasParserHelperTypes.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  //
  //  Definition of Basic Types
  //
  // ////////////////////////////////////////////////////////////////////
  // The types of iterator, scanner and rule are then derived from
  // the parsing unit.
  typedef std::istreambuf_iterator<char> base_iterator_t;
  typedef boost::spirit::multi_pass<base_iterator_t> iterator_t;

  // ////////////////////////////////////////////////////////////////////
  //
  //  Parser related types
  //
  // ////////////////////////////////////////////////////////////////////
  typedef boost::spirit::qi::int_parser<unsigned int, 10, 1, 1> int1_p_t;

  typedef boost::spirit::qi::uint_parser<int, 10, 2, 2> uint2_p_t;

  typedef boost::spirit::qi::uint_parser<int, 10, 4, 4> uint4_p_t;

  typedef boost::spirit::qi::uint_parser<int, 10, 1, 4> uint1_4_p_t;

  typedef boost::spirit::qi::uint_parser<hour_t, 10, 2, 2> hour_p_t;
  typedef boost::spirit::qi::uint_parser<minute_t, 10, 2, 2> minute_p_t;
  typedef boost::spirit::qi::uint_parser<second_t, 10, 2, 2> second_p_t;
  typedef boost::spirit::qi::uint_parser<year_t, 10, 4, 4> year_p_t;
  typedef boost::spirit::qi::uint_parser<month_t, 10, 2, 2> month_p_t;
  typedef boost::spirit::qi::uint_parser<day_t, 10, 2, 2> day_p_t;
}
#endif // __STDAIR_BAS_BASCOMPARSERTYPES_HPP
```

33.61 stdair/basic/BasTypes.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.62 stdair/basic/BasTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BAS_BASTYPES_HPP
#define __STDAIR_BAS_BASTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

}
#endif // __STDAIR_BAS_BASTYPES_HPP
```

33.63 stdair/basic/ContinuousAttributeLite.hpp File Reference

```
#include <cassert>
#include <iosfwd>
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DictionaryManager.hpp>
```

Classes

- struct stdair::ContinuousAttributeLite< T >

  Class modeling the distribution of values that can be taken by a continuous attribute.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.64 stdair/basic/ContinuousAttributeLite.hpp

```
#ifndef __STDAIR_BAS_CONTINUOUSATTRIBUTELITE_HPP
#define __STDAIR_BAS_CONTINUOUSATTRIBUTELITE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <iosfwd>
#include <string>
#include <vector>
#include <map>
// StdAir
#include <stdair/stdair_basic_types.hpp>
// TraDemGen
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DictionaryManager.hpp>

namespace stdair {

  template <typename T>
  struct ContinuousAttributeLite {
  public:
    // ////////////////////// Type definitions ////////////////////////
    typedef std::map<T, stdair::Probability_T> ContinuousDistribution_T;

  public:
    // ////////////////////// Business Methods ////////////////////
    const T getValue(const stdair::Probability_T& iCumulativeProbability) const{
      const DictionaryKey_T& lKey =
        DictionaryManager::valueToKey (iCumulativeProbability);

      // Find the first cumulative probablity value greater or equal to lKey.
      unsigned int idx = 0;
      for (; idx < _size; ++idx) {
        if (_cumulativeDistribution.at(idx) > lKey) {
          break;
        }
      }

      if (idx == 0) {
        return _valueArray.at(idx);
      }
      if (idx == _size) {
        return _valueArray.at(idx-1);
      }

      //
      const stdair::Probability_T& lCumulativeCurrentPoint =
        DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
      const T& lValueCurrentPoint = _valueArray.at(idx);

      //
      const stdair::Probability_T& lCumulativePreviousPoint =
        DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
      const T& lValuePreviousPoint = _valueArray.at(idx-1);

      if (lCumulativePreviousPoint == lCumulativeCurrentPoint) {
        return lValuePreviousPoint;
      }

      T oValue= lValuePreviousPoint + (lValueCurrentPoint - lValuePreviousPoint)
        * (iCumulativeProbability - lCumulativePreviousPoint)
        / (lCumulativeCurrentPoint - lCumulativePreviousPoint);

```

```
      return oValue;
    }

  public:
    // ////////////////////// Business Methods ////////////////////
    const stdair::Probability_T getRemainingProportion(const T& iValue) const {

      // Find the first value greater than iValue.
      unsigned int idx = 0;
      for (; idx < _size; ++idx) {
        if (_valueArray.at(idx) > iValue) {
          break;
        }
      }
      if (idx == 0) {
        const stdair::Probability_T& oCumulativeProbability =
          DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
        return 1 - oCumulativeProbability;
      }
      if (idx == _size) {
        const stdair::Probability_T& oCumulativeProbability =
          DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
        return 1 - oCumulativeProbability;
      }

      //
      const stdair::Probability_T& lCumulativeCurrentPoint =
        DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
      const T& lValueCurrentPoint = _valueArray.at(idx);

      //
      const stdair::Probability_T& lCumulativePreviousPoint =
        DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
      const T& lValuePreviousPoint = _valueArray.at(idx-1);

      if (lValuePreviousPoint == lValueCurrentPoint) {
        return 1 - lCumulativePreviousPoint;
      }

      const stdair::Probability_T& oCumulativeProbability =
        lCumulativePreviousPoint + (lCumulativeCurrentPoint - lCumulativePrevious
    Point)
        * (iValue - lValuePreviousPoint)
        / (lValueCurrentPoint - lValuePreviousPoint);

      return 1 - oCumulativeProbability;
    }

  public:
    // ////////////////////// Business Methods ////////////////////
    const double getDerivativeValue(const T iKey) const{

      // Find the first key value greater or equal to iKey.
      unsigned int idx = 0;
      for (; idx < _size; ++idx) {
        if (_valueArray.at(idx) > iKey) {
          break;
        }
      }
      assert (idx != 0);
      assert (idx != _size);

      //
      const stdair::Probability_T& lCumulativeCurrentPoint =
        DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
      const T& lValueCurrentPoint = _valueArray.at(idx);
```

```
00147
00148       //
00149       const stdair::Probability_T& lCumulativePreviousPoint =
00150         DictionaryManager::keyToValue (_cumulativeDistribution.at(idx-1));
00151       const T& lValuePreviousPoint = _valueArray.at(idx-1);
00152       assert (lValueCurrentPoint != lValuePreviousPoint);
00153
00154       const double oValue= (lCumulativeCurrentPoint - lCumulativePreviousPoint)
00155         / (lValueCurrentPoint - lValuePreviousPoint);
00156
00157       return oValue;
00158     }
00159
00163     const T getUpperBound (const T iKey) const {
00164       // Find the first key value greater or equal to iKey.
00165       unsigned int idx = 0;
00166       for (; idx < _size; ++idx) {
00167         if (_valueArray.at(idx) > iKey) {
00168           break;
00169         }
00170       }
00171       assert (idx != 0);
00172       assert (idx != _size);
00173
00174       return _valueArray.at (idx);
00175     }
00176
00177   public:
00178     // ############## Display Support Methods ################
00182     const std::string displayCumulativeDistribution() const {
00183       std::ostringstream oStr;
00184
00185       for (unsigned int idx = 0; idx < _size; ++idx) {
00186         if (idx != 0) {
00187           oStr << ", ";
00188         }
00189
00190         const stdair::Probability_T& lProbability =
00191           DictionaryManager::keyToValue (_cumulativeDistribution.at(idx));
00192
00193         oStr << _valueArray.at(idx) << ":" << lProbability;
00194       }
00195       return oStr.str();
00196     }
00197
00198
00199   public:
00200     // ########## Constructors and destructors ##############
00204     ContinuousAttributeLite (const ContinuousDistribution_T& iValueMap)
00205       : _size (iValueMap.size()) {
00206       init (iValueMap);
00207     }
00208
00212     ContinuousAttributeLite (const ContinuousAttributeLite& iCAL)
00213       : _size (iCAL._size),
00214         _cumulativeDistribution (iCAL._cumulativeDistribution),
00215         _valueArray (iCAL._valueArray) {
00216     }
00217
00221     ContinuousAttributeLite& operator= (const ContinuousAttributeLite& iCAL) {
00222       _size = iCAL._size;
00223       _cumulativeDistribution = iCAL._cumulativeDistribution;
00224       _valueArray = iCAL._valueArray;
00225       return *this;
00226     }
00227
00231     virtual ~ContinuousAttributeLite() {
```

```
    }

  private:
    ContinuousAttributeLite() : _size(1) {
    }

    void init (const ContinuousDistribution_T& iValueMap) {
      //
      const unsigned int lSize = iValueMap.size();
      _cumulativeDistribution.reserve (lSize);
      _valueArray.reserve (lSize);

      // Browse the map to retrieve the values and cumulative probabilities.
      for (typename ContinuousDistribution_T::const_iterator it =
             iValueMap.begin(); it != iValueMap.end(); ++it) {

        const T& attributeValue = it->first;
        const DictionaryKey_T& lKey = DictionaryManager::valueToKey (it->second);

        // Build the two arrays.
        _cumulativeDistribution.push_back (lKey);
        _valueArray.push_back (attributeValue);
      }
    }


  private:
    // ////////// Attributes //////////
    unsigned int _size;

    std::vector<DictionaryKey_T> _cumulativeDistribution;

    std::vector<T> _valueArray;
  };

}
#endif // __STDAIR_BAS_CONTINUOUSATTRIBUTELITE_HPP
```

33.65 stdair/basic/DemandGenerationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DemandGenerationMethod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.66 stdair/basic/DemandGenerationMethod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/DemandGenerationMethod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string DemandGenerationMethod::_labels[LAST_VALUE] =
    { "PoissonProcess", "StatisticsOrder" };

  // ////////////////////////////////////////////////////////////////////
  const char DemandGenerationMethod::_methodLabels[LAST_VALUE] = { 'P', 'S' };


  // ////////////////////////////////////////////////////////////////////
  DemandGenerationMethod::DemandGenerationMethod() : _method (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  DemandGenerationMethod::
  DemandGenerationMethod (const DemandGenerationMethod& iDemandGenerationMethod)
    : _method (iDemandGenerationMethod._method) {
  }

  // ////////////////////////////////////////////////////////////////////
  DemandGenerationMethod::
  DemandGenerationMethod (const EN_DemandGenerationMethod& iDemandGenerationMetho
d)
    : _method (iDemandGenerationMethod) {
  }

  // ////////////////////////////////////////////////////////////////////
  DemandGenerationMethod::EN_DemandGenerationMethod
  DemandGenerationMethod::getMethod (const char iMethodChar) {
    EN_DemandGenerationMethod oMethod;
    switch (iMethodChar) {
    case 'P': oMethod = POI_PRO; break;
    case 'S': oMethod = STA_ORD; break;
    default: oMethod = LAST_VALUE; break;
    }

    if (oMethod == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The demand (booking request) generation method '"
               << iMethodChar
               << "' is not known. Known demand (booking request) generation "
               << "methods: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }

    return oMethod;
  }

  // ////////////////////////////////////////////////////////////////////
  DemandGenerationMethod::DemandGenerationMethod (const char iMethodChar)
    : _method (getMethod (iMethodChar)) {
  }
```

```

// ////////////////////////////////////////////////////////////////////
DemandGenerationMethod::
DemandGenerationMethod (const std::string& iMethodStr) {
  //
  const size_t lSize = iMethodStr.size();
  assert (lSize == 1);
  const char lMethodChar = iMethodStr[0];
  _method = getMethod (lMethodChar);
}

// ////////////////////////////////////////////////////////////////////
const std::string& DemandGenerationMethod::
getLabel (const EN_DemandGenerationMethod& iMethod) {
  return _labels[iMethod];
}

// ////////////////////////////////////////////////////////////////////
char DemandGenerationMethod::
getMethodLabel (const EN_DemandGenerationMethod& iMethod) {
  return _methodLabels[iMethod];
}

// ////////////////////////////////////////////////////////////////////
std::string DemandGenerationMethod::
getMethodLabelAsString (const EN_DemandGenerationMethod& iMethod) {
  std::ostringstream oStr;
  oStr << _methodLabels[iMethod];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
std::string DemandGenerationMethod::describeLabels() {
  std::ostringstream ostr;
  for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
    if (idx != 0) {
      ostr << ", ";
    }
    ostr << _labels[idx];
  }
  return ostr.str();
}

// ////////////////////////////////////////////////////////////////////
DemandGenerationMethod::EN_DemandGenerationMethod
DemandGenerationMethod::getMethod() const {
  return _method;
}

// ////////////////////////////////////////////////////////////////////
char DemandGenerationMethod::getMethodAsChar() const {
  const char oMethodChar = _methodLabels[_method];
  return oMethodChar;
}

// ////////////////////////////////////////////////////////////////////
std::string DemandGenerationMethod::getMethodAsString() const {
  std::ostringstream oStr;
  oStr << _methodLabels[_method];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string DemandGenerationMethod::describe() const {
  std::ostringstream ostr;
  ostr << _labels[_method];
  return ostr.str();
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  bool DemandGenerationMethod::
  operator== (const EN_DemandGenerationMethod& iMethod) const {
    return (_method == iMethod);
  }

}
```

33.67 stdair/basic/DemandGenerationMethod.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::DemandGenerationMethod

  Enumeration of demand (booking request) generation methods.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.68 stdair/basic/DemandGenerationMethod.hpp

```
#ifndef __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
#define __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct DemandGenerationMethod : public StructAbstract {
  public:
    typedef enum {
      POI_PRO = 0,
      STA_ORD,
      LAST_VALUE
    } EN_DemandGenerationMethod;

    static const std::string& getLabel (const EN_DemandGenerationMethod&);

    static EN_DemandGenerationMethod getMethod (const char);

    static char getMethodLabel (const EN_DemandGenerationMethod&);

    static std::string getMethodLabelAsString (const EN_DemandGenerationMethod&);

    static std::string describeLabels();

    EN_DemandGenerationMethod getMethod() const;

    char getMethodAsChar() const;

    std::string getMethodAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_DemandGenerationMethod&) const;

  public:
    DemandGenerationMethod (const EN_DemandGenerationMethod&);
    DemandGenerationMethod (const char iMethod);
    DemandGenerationMethod (const std::string& iMethod);
    DemandGenerationMethod (const DemandGenerationMethod&);

  private:
    DemandGenerationMethod();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _methodLabels[LAST_VALUE];

  private:
    // //////// Attributes /////////
    EN_DemandGenerationMethod _method;
  };

}
#endif // __STDAIR_BAS_DEMANDGENERATIONMETHOD_HPP
```

33.69 stdair/basic/DictionaryManager.cpp File Reference

```
#include <stdair/basic/DictionaryManager.hpp>
#include <stdair/basic/BasConst_General.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.70 stdair/basic/DictionaryManager.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/basic/DictionaryManager.hpp>
#include <stdair/basic/BasConst_General.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const stdair::Probability_T DictionaryManager::
  keyToValue (const DictionaryKey_T iKey) {
    const float lValue =
      static_cast<float> (iKey) / DEFAULT_NUMBER_OF_SUBDIVISIONS;
    const stdair::Probability_T lProbability (lValue);
    return lProbability;
  }

  // ////////////////////////////////////////////////////////////////////
  const DictionaryKey_T DictionaryManager::
  valueToKey (const stdair::Probability_T iValue) {
    const unsigned short lValueMultipliedByThousand =
      static_cast<unsigned short> (iValue) * DEFAULT_NUMBER_OF_SUBDIVISIONS;
    const DictionaryKey_T lDictionaryKey (lValueMultipliedByThousand);
    return lDictionaryKey;
  }

}
```

33.71 stdair/basic/DictionaryManager.hpp File Reference

```
#include <stdair/stdair_maths_types.hpp>
```

Classes

- class stdair::DictionaryManager

  Class wrapper of dictionary business methods.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef unsigned short stdair::DictionaryKey_T

33.72 stdair/basic/DictionaryManager.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00003 #define __STDAIR_BASIC_DICTIONARYMANAGER_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // StdAir
00009 #include <stdair/stdair_maths_types.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////// Type definitions /////////////////
00017   typedef unsigned short DictionaryKey_T;
00018
00022   class DictionaryManager {
00023   public:
00024     // ////////////// Business methods /////////////////
00028     static const stdair::Probability_T keyToValue (const DictionaryKey_T);
00029
00033     static const DictionaryKey_T valueToKey (const stdair::Probability_T);
00034   };
00035 }
00036 #endif // __STDAIR_BASIC_DICTIONARYMANAGER_HPP
```

33.73 stdair/basic/EventType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/EventType.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.74 stdair/basic/EventType.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/EventType.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string EventType::_labels[LAST_VALUE] =
00015     { "BookingRequest", "Cancellation","OptimisationNotificationForFlightDate",
00016       "OptimisationNotificationForNetwork", "ScheduleChange", "Snapshot",
00017       "RevenueManagement", "BreakPoint" };
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   const char EventType::
00021   _typeLabels[LAST_VALUE] = { 'B', 'X', 'F', 'N', 'C', 'S', 'R', 'P' };
00022
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   EventType::EventType()
00026     : _type (LAST_VALUE) {
00027     assert (false);
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   EventType::EventType (const EventType& iEventType)
00032     : _type (iEventType._type) {
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   EventType::EventType (const EN_EventType& iEventType)
00037     : _type (iEventType) {
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   EventType::EventType (const char iType) {
00042     switch (iType) {
00043     case 'B': _type = BKG_REQ; break;
00044     case 'X': _type = CX; break;
00045     case 'F': _type = OPT_NOT_4_FD; break;
00046     case 'N': _type = OPT_NOT_4_NET; break;
00047     case 'C': _type = SKD_CHG; break;
00048     case 'S': _type = SNAPSHOT; break;
00049     case 'R': _type = RM; break;
00050     case 'P': _type = BRK_PT; break;
00051     default: _type = LAST_VALUE; break;
00052     }
00053
00054     if (_type == LAST_VALUE) {
00055       const std::string& lLabels = describeLabels();
00056       std::ostringstream oMessage;
00057       oMessage << "The event type '" << iType
00058                << "' is not known. Known event types: " << lLabels;
00059       throw CodeConversionException (oMessage.str());
00060     }
00061   }
00062
00063   // ////////////////////////////////////////////////////////////////////
00064   EventType::EventType (const std::string& iTypeStr) {
00065     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
```

```
      if (iTypeStr.compare(_labels[idx]) == 0) {
        _type = static_cast<EN_EventType> (idx);
        break;
      } else {
        _type = LAST_VALUE;
      }
    }
    if (_type == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The event type '" << iTypeStr
               << "' is not known. Known event types: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& EventType::getLabel (const EN_EventType& iType) {
    return _labels[iType];
  }

  // ////////////////////////////////////////////////////////////////////
  char EventType::getTypeLabel (const EN_EventType& iType) {
    return _typeLabels[iType];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string EventType::getTypeLabelAsString (const EN_EventType& iType) {
    std::ostringstream oStr;
    oStr << _typeLabels[iType];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string EventType::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx];
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  EventType::EN_EventType EventType::getType() const {
    return _type;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string EventType::getTypeAsString() const {
    std::ostringstream oStr;
    oStr << _typeLabels[_type];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string EventType::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_type];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool EventType::operator== (const EN_EventType& iType) const {
    return (_type == iType);
```

```
  }

}
```

33.75 stdair/basic/EventType.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::EventType

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.76 stdair/basic/EventType.hpp

```
00001 #ifndef __STDAIR_BAS_EVENTTYPE_HPP
00002 #define __STDAIR_BAS_EVENTTYPE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00015   struct EventType : public StructAbstract {
00016   public:
00017     typedef enum {
00018       BKG_REQ = 0,
00019       CX,
00020       OPT_NOT_4_FD,
00021       OPT_NOT_4_NET,
00022       SKD_CHG,
00023       SNAPSHOT,
00024       RM,
00025       BRK_PT,
00026       LAST_VALUE
00027     } EN_EventType;
00028
00034     static const std::string& getLabel (const EN_EventType&);
00035
00039     static char getTypeLabel (const EN_EventType&);
00040
00044     static std::string getTypeLabelAsString (const EN_EventType&);
00045
00047     static std::string describeLabels();
00048
00050     EN_EventType getType() const;
00051
00055     std::string getTypeAsString() const;
00056
00062     const std::string describe() const;
00063
00064   public:
00066     bool operator== (const EN_EventType&) const;
00067
00068   public:
00070     EventType (const EN_EventType&);
00072     EventType (const char iType);
00074     EventType (const std::string& iTypeStr);
00076     EventType (const EventType&);
00077
00078   private:
00080     EventType();
00081
00082
00083   private:
00085     static const std::string _labels[LAST_VALUE];
00087     static const char _typeLabels[LAST_VALUE];
00088
00089
00090   private:
00091     // //////// Attributes /////////
00093     EN_EventType _type;
00094   };
00095
00096 }
```

```
00097 #endif // __STDAIR_BAS_EVENTTYPE_HPP
```

33.77 stdair/basic/float_utils.hpp File Reference

```
#include <stdair/basic/float_utils_google.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.78 stdair/basic/float_utils.hpp

```
#ifndef __STDAIR_BAS_FLOAT_UTILS_HPP
#define __STDAIR_BAS_FLOAT_UTILS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/basic/float_utils_google.hpp>

namespace stdair {

}
#endif // __STDAIR_BAS_FLOAT_UTILS_HPP
```

33.79 stdair/basic/float_utils_google.hpp File Reference

Classes

- class TypeWithSize< size >
- class TypeWithSize< 4 >
- class TypeWithSize< 8 >
- class FloatingPoint< RawType >

33.80 stdair/basic/float_utils_google.hpp

```
#ifndef __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
#define __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP

// Redistribution and use in source and binary forms, with or without
// modification, are permitted provided that the following conditions are
// met:
//
//     * Redistributions of source code must retain the above copyright
// notice, this list of conditions and the following disclaimer.
//     * Redistributions in binary form must reproduce the above
// copyright notice, this list of conditions and the following disclaimer
// in the documentation and/or other materials provided with the
// distribution.
//     * Neither the name of Google Inc. nor the names of its
// contributors may be used to endorse or promote products derived from
// this software without specific prior written permission.
//
// THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
// "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
// LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
// A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
// OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
// SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
// LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
// DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
// THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
// (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
// OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
//
// Authors: wan@google.com (Zhanyong Wan), eefacm@gmail.com (Sean Mcafee)
//
// The Google C++ Testing Framework (Google Test)


// This template class serves as a compile-time function from size to
// type.  It maps a size in bytes to a primitive type with that
// size. e.g.
//
//   TypeWithSize<4>::UInt
//
// is typedef-ed to be unsigned int (unsigned integer made up of 4
// bytes).
//
// Such functionality should belong to STL, but I cannot find it
// there.
//
// Google Test uses this class in the implementation of floating-point
// comparison.
//
// For now it only handles UInt (unsigned int) as that's all Google Test
// needs.  Other types can be easily added in the future if need
// arises.
template <size_t size>
class TypeWithSize {
 public:
  // This prevents the user from using TypeWithSize<N> with incorrect
  // values of N.
  typedef void UInt;
};

// The specialization for size 4.
template <>
class TypeWithSize<4> {
 public:
  // unsigned int has size 4 in both gcc and MSVC.
```

```
  //
  // As base/basictypes.h doesn't compile on Windows, we cannot use
  // uint32, uint64, and etc here.
  typedef int Int;
  typedef unsigned int UInt;
};

// The specialization for size 8.
template <>
class TypeWithSize<8> {
 public:
#if GTEST_OS_WINDOWS
  typedef __int64 Int;
  typedef unsigned __int64 UInt;
#else
  typedef long long Int;  // NOLINT
  typedef unsigned long long UInt;  // NOLINT
#endif  // GTEST_OS_WINDOWS
};


// This template class represents an IEEE floating-point number
// (either single-precision or double-precision, depending on the
// template parameters).
//
// The purpose of this class is to do more sophisticated number
// comparison.  (Due to round-off error, etc, it's very unlikely that
// two floating-points will be equal exactly.  Hence a naive
// comparison by the == operation often doesn't work.)
//
// Format of IEEE floating-point:
//
//   The most-significant bit being the leftmost, an IEEE
//   floating-point looks like
//
//     sign_bit exponent_bits fraction_bits
//
//   Here, sign_bit is a single bit that designates the sign of the
//   number.
//
//   For float, there are 8 exponent bits and 23 fraction bits.
//
//   For double, there are 11 exponent bits and 52 fraction bits.
//
//   More details can be found at
//   http://en.wikipedia.org/wiki/IEEE_floating-point_standard.
//
// Template parameter:
//
//   RawType: the raw floating-point type (either float or double)
template <typename RawType>
class FloatingPoint {
 public:
  // Defines the unsigned integer type that has the same size as the
  // floating point number.
  typedef typename TypeWithSize<sizeof(RawType)>::UInt Bits;

  // Constants.

  // # of bits in a number.
  static const size_t kBitCount = 8*sizeof(RawType);

  // # of fraction bits in a number.
  static const size_t kFractionBitCount =
    std::numeric_limits<RawType>::digits - 1;

  // # of exponent bits in a number.
```

```
  static const size_t kExponentBitCount = kBitCount - 1 - kFractionBitCount;

  // The mask for the sign bit.
  static const Bits kSignBitMask = static_cast<Bits>(1) << (kBitCount - 1);

  // The mask for the fraction bits.
  static const Bits kFractionBitMask =
    ~static_cast<Bits>(0) >> (kExponentBitCount + 1);

  // The mask for the exponent bits.
  static const Bits kExponentBitMask = ~(kSignBitMask | kFractionBitMask);

  // How many ULP's (Units in the Last Place) we want to tolerate when
  // comparing two numbers.  The larger the value, the more error we
  // allow.  A 0 value means that two numbers must be exactly the same
  // to be considered equal.
  //
  // The maximum error of a single floating-point operation is 0.5
  // units in the last place.  On Intel CPU's, all floating-point
  // calculations are done with 80-bit precision, while double has 64
  // bits.  Therefore, 4 should be enough for ordinary use.
  //
  // See the following article for more details on ULP:
  // http://www.cygnus-software.com/papers/comparingfloats/comparingfloats.htm.
  static const size_t kMaxUlps = 4;

  // Constructs a FloatingPoint from a raw floating-point number.
  //
  // On an Intel CPU, passing a non-normalized NAN (Not a Number)
  // around may change its bits, although the new value is guaranteed
  // to be also a NAN.  Therefore, don't expect this constructor to
  // preserve the bits in x when x is a NAN.
  explicit FloatingPoint(const RawType& x) { u_.value_ = x; }

  // Static methods

  // Reinterprets a bit pattern as a floating-point number.
  //
  // This function is needed to test the AlmostEquals() method.
  static RawType ReinterpretBits(const Bits bits) {
    FloatingPoint fp(0);
    fp.u_.bits_ = bits;
    return fp.u_.value_;
  }

  // Returns the floating-point number that represent positive infinity.
  static RawType Infinity() {
    return ReinterpretBits(kExponentBitMask);
  }

  // Non-static methods

  // Returns the bits that represents this number.
  const Bits &bits() const { return u_.bits_; }

  // Returns the exponent bits of this number.
  Bits exponent_bits() const { return kExponentBitMask & u_.bits_; }

  // Returns the fraction bits of this number.
  Bits fraction_bits() const { return kFractionBitMask & u_.bits_; }

  // Returns the sign bit of this number.
  Bits sign_bit() const { return kSignBitMask & u_.bits_; }

  // Returns true iff this is NAN (not a number).
  bool is_nan() const {
    // It's a NAN if the exponent bits are all ones and the fraction
```

```
    // bits are not entirely zeros.
    return (exponent_bits() == kExponentBitMask) && (fraction_bits() != 0);
  }

  // Returns true iff this number is at most kMaxUlps ULP's away from
  // rhs.  In particular, this function:
  //
  //   - returns false if either number is (or both are) NAN.
  //   - treats really large numbers as almost equal to infinity.
  //   - thinks +0.0 and -0.0 are 0 DLP's apart.
  bool AlmostEquals(const FloatingPoint& rhs) const {
    // The IEEE standard says that any comparison operation involving
    // a NAN must return false.
    if (is_nan() || rhs.is_nan()) return false;

    return DistanceBetweenSignAndMagnitudeNumbers(u_.bits_, rhs.u_.bits_)
        <= kMaxUlps;
  }

 private:
  // The data type used to store the actual floating-point number.
  union FloatingPointUnion {
    RawType value_;  // The raw floating-point number.
    Bits bits_;      // The bits that represent the number.
  };

  // Converts an integer from the sign-and-magnitude representation to
  // the biased representation.  More precisely, let N be 2 to the
  // power of (kBitCount - 1), an integer x is represented by the
  // unsigned number x + N.
  //
  // For instance,
  //
  //   -N + 1 (the most negative number representable using
  //          sign-and-magnitude) is represented by 1;
  //   0      is represented by N; and
  //   N - 1  (the biggest number representable using
  //          sign-and-magnitude) is represented by 2N - 1.
  //
  // Read http://en.wikipedia.org/wiki/Signed_number_representations
  // for more details on signed number representations.
  static Bits SignAndMagnitudeToBiased(const Bits &sam) {
    if (kSignBitMask & sam) {
      // sam represents a negative number.
      return ~sam + 1;
    } else {
      // sam represents a positive number.
      return kSignBitMask | sam;
    }
  }

  // Given two numbers in the sign-and-magnitude representation,
  // returns the distance between them as an unsigned number.
  static Bits DistanceBetweenSignAndMagnitudeNumbers(const Bits &sam1,
                                                     const Bits &sam2) {
    const Bits biased1 = SignAndMagnitudeToBiased(sam1);
    const Bits biased2 = SignAndMagnitudeToBiased(sam2);
    return (biased1 >= biased2) ? (biased1 - biased2) : (biased2 - biased1);
  }

  FloatingPointUnion u_;
};

#endif // __STDAIR_BAS_FLOAT_UTILS_GOOGLE_HPP
```

33.81 stdair/basic/ForecastingMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.82 stdair/basic/ForecastingMethod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string ForecastingMethod::_labels[LAST_VALUE] =
    { "Q Forecasting", "Hybrid Forecasting", "Old QFF", "New QFF",
      "Based Forecasting" };

  // ////////////////////////////////////////////////////////////////////
  const char ForecastingMethod::
  _methodLabels[LAST_VALUE] = { 'Q', 'H', 'O' , 'N', 'B'};


  // ////////////////////////////////////////////////////////////////////
  ForecastingMethod::ForecastingMethod()
    : _method (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  ForecastingMethod::
  ForecastingMethod (const ForecastingMethod& iForecastingMethod)
    : _method (iForecastingMethod._method) {
  }

  // ////////////////////////////////////////////////////////////////////
  ForecastingMethod::
  ForecastingMethod (const EN_ForecastingMethod& iForecastingMethod)
    : _method (iForecastingMethod) {
  }

  // ////////////////////////////////////////////////////////////////////
  ForecastingMethod::ForecastingMethod (const char iMethod) {
    switch (iMethod) {
    case 'Q': _method = Q_FORECASTING; break;
    case 'H': _method = HYBRID_FORECASTING; break;
    case 'O': _method = OLD_QFF; break;
    case 'N': _method = NEW_QFF; break;
    case 'B': _method = BASED_FORECASTING; break;
    default: _method = LAST_VALUE; break;
    }

    if (_method == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The forecasting method '" << iMethod
               << "' is not known. Known forecasting methods: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& ForecastingMethod::
  getLabel (const EN_ForecastingMethod& iMethod) {
    return _labels[iMethod];
  }
```

```cpp

// ////////////////////////////////////////////////////////////////////
char ForecastingMethod::getMethodLabel (const EN_ForecastingMethod& iMethod) {
  return _methodLabels[iMethod];
}

// ////////////////////////////////////////////////////////////////////
std::string ForecastingMethod::
getMethodLabelAsString (const EN_ForecastingMethod& iMethod) {
  std::ostringstream oStr;
  oStr << _methodLabels[iMethod];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
std::string ForecastingMethod::describeLabels() {
  std::ostringstream ostr;
  for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
    if (idx != 0) {
      ostr << ", ";
    }
    ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
  }
  return ostr.str();
}

// ////////////////////////////////////////////////////////////////////
ForecastingMethod::EN_ForecastingMethod ForecastingMethod::getMethod() const {
  return _method;
}

// ////////////////////////////////////////////////////////////////////
std::string ForecastingMethod::getMethodAsString() const {
  std::ostringstream oStr;
  oStr << _methodLabels[_method];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string ForecastingMethod::describe() const {
  std::ostringstream ostr;
  ostr << _labels[_method];
  return ostr.str();
}

// ////////////////////////////////////////////////////////////////////
bool ForecastingMethod::
operator== (const EN_ForecastingMethod& iMethod) const {
  return (_method == iMethod);
}

}
```

33.83 stdair/basic/ForecastingMethod.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::ForecastingMethod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.84 stdair/basic/ForecastingMethod.hpp

```
#ifndef __STDAIR_BAS_FORECASTINGMETHOD_HPP
#define __STDAIR_BAS_FORECASTINGMETHOD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct ForecastingMethod : public StructAbstract {
  public:
    typedef enum {
      Q_FORECASTING = 0,
      HYBRID_FORECASTING,
      OLD_QFF,
      NEW_QFF,
      BASED_FORECASTING,
      LAST_VALUE
    } EN_ForecastingMethod;

    static const std::string& getLabel (const EN_ForecastingMethod&);

    static char getMethodLabel (const EN_ForecastingMethod&);

    static std::string getMethodLabelAsString (const EN_ForecastingMethod&);

    static std::string describeLabels();

    EN_ForecastingMethod getMethod() const;

    std::string getMethodAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_ForecastingMethod&) const;

  public:
    ForecastingMethod (const EN_ForecastingMethod&);
    ForecastingMethod (const char iMethod);
    ForecastingMethod (const ForecastingMethod&);

  private:
    ForecastingMethod();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _methodLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_ForecastingMethod _method;
  };

}
#endif // __STDAIR_BAS_FORECASTINGMETHOD_HPP
```

33.85 stdair/basic/JSonCommand.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/JSonCommand.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.86 stdair/basic/JSonCommand.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/JSonCommand.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string JSonCommand::_labels[LAST_VALUE] =
    { "list", "flight_date", "event_list", "break_point", "run", "reset", "status", "config"};

  // ////////////////////////////////////////////////////////////////////
  JSonCommand::JSonCommand()
    : _command (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  JSonCommand::
  JSonCommand (const JSonCommand& iJSonCommand)
    : _command (iJSonCommand._command) {
  }

  // ////////////////////////////////////////////////////////////////////
  JSonCommand::EN_JSonCommand
  JSonCommand::getCommand (const std::string& iCommandStr) {

    EN_JSonCommand oJSonCommand;
    if (iCommandStr == "list") {
      oJSonCommand = LIST;
    } else if (iCommandStr == "flight_date") {
      oJSonCommand = FLIGHT_DATE;
    } else if (iCommandStr == "event_list") {
      oJSonCommand = EVENT_LIST;
    } else if (iCommandStr == "break_point") {
      oJSonCommand = BREAK_POINT;
    } else if (iCommandStr == "run") {
      oJSonCommand = RUN;
    } else if (iCommandStr == "reset") {
      oJSonCommand = RESET;
    } else if (iCommandStr == "status") {
      oJSonCommand = STATUS;
    } else if (iCommandStr == "config") {
      oJSonCommand = CONFIG;
    } else {
      oJSonCommand = LAST_VALUE;
    }

    if (oJSonCommand == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The JSON command '" << iCommandStr
               << "' is not known. Known forecasting commands: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }

    return oJSonCommand;
  }

```

```
00065   // ////////////////////////////////////////////////////////////////////
00066   std::string JSonCommand::getLabel(const EN_JSonCommand& iCommand) {
00067     return _labels[iCommand];
00068   }
00069
00070   // ////////////////////////////////////////////////////////////////////
00071   JSonCommand::JSonCommand (const std::string& iCommandStr) {
00072     //
00073     _command = getCommand (iCommandStr);
00074   }
00075
00076   // ////////////////////////////////////////////////////////////////////
00077   std::string JSonCommand::describeLabels() {
00078     std::ostringstream ostr;
00079     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00080       if (idx != 0) {
00081         ostr << ", ";
00082       }
00083       ostr << _labels[idx] << " ";//'" << _commandLabels[idx] << "'";
00084     }
00085     return ostr.str();
00086   }
00087
00088   // ////////////////////////////////////////////////////////////////////
00089   JSonCommand::EN_JSonCommand JSonCommand::getCommand() const {
00090     return _command;
00091   }
00092
00093   // ////////////////////////////////////////////////////////////////////
00094   const std::string JSonCommand::describe() const {
00095     std::ostringstream ostr;
00096     ostr << _labels[_command];
00097     return ostr.str();
00098   }
00099
00100   // ////////////////////////////////////////////////////////////////////
00101   bool JSonCommand::
00102   operator== (const EN_JSonCommand& iCommand) const {
00103     return (_command == iCommand);
00104   }
00105
00106 }
```

33.87 stdair/basic/JSonCommand.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::JSonCommand

  Enumeration of json commands.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.88 stdair/basic/JSonCommand.hpp

```
#ifndef __STDAIR_BAS_JSONCOMMAND_HPP
#define __STDAIR_BAS_JSONCOMMAND_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct JSonCommand : public StructAbstract {
  public:
    typedef enum {
      LIST = 0,
      FLIGHT_DATE,
      EVENT_LIST,
      BREAK_POINT,
      RUN,
      RESET,
      STATUS,
      CONFIG,
      LAST_VALUE
    } EN_JSonCommand;

    static EN_JSonCommand getCommand (const std::string& iCommandStr);

    static std::string getLabel(const EN_JSonCommand&);

    static std::string describeLabels();

    EN_JSonCommand getCommand() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_JSonCommand&) const;

  public:
    JSonCommand (const EN_JSonCommand&);

    JSonCommand (const std::string&);

    JSonCommand (const JSonCommand&);

  private:
    JSonCommand();


  private:
    static const std::string _labels[LAST_VALUE];

  private:
    // //////// Attributes /////////
    EN_JSonCommand _command;
  };

}
#endif // __STDAIR_BAS_JSONCOMMAND_HPP
```

33.89 stdair/basic/OptimisationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.90 stdair/basic/OptimisationMethod.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/OptimisationMethod.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string OptimisationMethod::_labels[LAST_VALUE] =
00015     { "Leg based Monte Carlo", "Leg based EMSRb"};
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   const char OptimisationMethod::
00019   _methodLabels[LAST_VALUE] = { 'M', 'E'};
00020
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   OptimisationMethod::OptimisationMethod()
00024     : _method (LAST_VALUE) {
00025     assert (false);
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   OptimisationMethod::
00030   OptimisationMethod (const OptimisationMethod& iOptimisationMethod)
00031     : _method (iOptimisationMethod._method) {
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   OptimisationMethod::
00036   OptimisationMethod (const EN_OptimisationMethod& iOptimisationMethod)
00037     : _method (iOptimisationMethod) {
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   OptimisationMethod::OptimisationMethod (const char iMethod) {
00042     switch (iMethod) {
00043     case 'M': _method = LEG_BASED_MC; break;
00044     case 'E': _method = LEG_BASED_EMSR_B; break;
00045     default: _method = LAST_VALUE; break;
00046     }
00047
00048     if (_method == LAST_VALUE) {
00049       const std::string& lLabels = describeLabels();
00050       std::ostringstream oMessage;
00051       oMessage << "The optimisation method '" << iMethod
00052                << "' is not known. Known optimisation methods: " << lLabels;
00053       throw CodeConversionException (oMessage.str());
00054     }
00055   }
00056
00057   // ////////////////////////////////////////////////////////////////////
00058   const std::string& OptimisationMethod::
00059   getLabel (const EN_OptimisationMethod& iMethod) {
00060     return _labels[iMethod];
00061   }
00062
00063   // ////////////////////////////////////////////////////////////////////
00064   char OptimisationMethod::getMethodLabel (const EN_OptimisationMethod& iMethod)
      {
```

```
    return _methodLabels[iMethod];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string OptimisationMethod::
  getMethodLabelAsString (const EN_OptimisationMethod& iMethod) {
    std::ostringstream oStr;
    oStr << _methodLabels[iMethod];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string OptimisationMethod::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  OptimisationMethod::EN_OptimisationMethod OptimisationMethod::getMethod() const
  {
    return _method;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string OptimisationMethod::getMethodAsString() const {
    std::ostringstream oStr;
    oStr << _methodLabels[_method];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string OptimisationMethod::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_method];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool OptimisationMethod::
  operator== (const EN_OptimisationMethod& iMethod) const {
    return (_method == iMethod);
  }

}
```

33.91 stdair/basic/OptimisationMethod.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::OptimisationMethod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.92 stdair/basic/OptimisationMethod.hpp

```
#ifndef __STDAIR_BAS_OPTIMISATIONMETHOD_HPP
#define __STDAIR_BAS_OPTIMISATIONMETHOD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct OptimisationMethod : public StructAbstract {
  public:
    typedef enum {
      LEG_BASED_MC = 0,
      LEG_BASED_EMSR_B,
      LAST_VALUE
    } EN_OptimisationMethod;

    static const std::string& getLabel (const EN_OptimisationMethod&);

    static char getMethodLabel (const EN_OptimisationMethod&);

    static std::string getMethodLabelAsString (const EN_OptimisationMethod&);

    static std::string describeLabels();

    EN_OptimisationMethod getMethod() const;

    std::string getMethodAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_OptimisationMethod&) const;

  public:
    OptimisationMethod (const EN_OptimisationMethod&);
    OptimisationMethod (const char iMethod);
    OptimisationMethod (const OptimisationMethod&);

  private:
    OptimisationMethod();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _methodLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_OptimisationMethod _method;
  };

}
#endif // __STDAIR_BAS_OPTIMISATIONMETHOD_HPP
```

33.93 stdair/basic/PartnershipTechnique.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.94 stdair/basic/PartnershipTechnique.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string PartnershipTechnique::_labels[LAST_VALUE] =
    { "None",
      "RevenueAvailabilityExchangeDemandAggregation",
      "RevenueAvailabilityExchangeYieldProration",
      "InterlineBidPriceDemandAggregation",
      "InterlineBidPriceYieldProration",
      "NonProtectionistInterlineBidPriceYieldProration",
      "RevenueManagementCooperation",
      "AdvancedRevenueManagementCooperation"};

  // ////////////////////////////////////////////////////////////////////
  const char PartnershipTechnique::_techniqueLabels[LAST_VALUE] = { 'N',
                                                                    'r',
                                                                    'R',
                                                                    'i',
                                                                    'I',
                                                                    'U',
                                                                    'C',
                                                                    'A'};


  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::PartnershipTechnique() : _technique (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::
  PartnershipTechnique (const PartnershipTechnique& iPartnershipTechnique)
    : _technique (iPartnershipTechnique._technique) {
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::
  PartnershipTechnique (const EN_PartnershipTechnique& iPartnershipTechnique)
    : _technique (iPartnershipTechnique) {
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::EN_PartnershipTechnique
  PartnershipTechnique::getTechnique (const char iTechniqueChar) {
    EN_PartnershipTechnique oTechnique;
    switch (iTechniqueChar) {
    case 'N': oTechnique = NONE; break;
    case 'r': oTechnique = RAE_DA; break;
    case 'R': oTechnique = RAE_YP; break;
    case 'i': oTechnique = IBP_DA; break;
    case 'I': oTechnique = IBP_YP; break;
    case 'U': oTechnique = IBP_YP_U; break;
    case 'C': oTechnique = RMC; break;
    case 'A': oTechnique = A_RMC; break;
    default: oTechnique = LAST_VALUE; break;
```

```
    }

    if (oTechnique == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The partnership technique '"
               << iTechniqueChar
               << "' is not known. Known partnership techniques: "
               << lLabels;
      throw CodeConversionException (oMessage.str());
    }

    return oTechnique;
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::PartnershipTechnique (const char iTechniqueChar)
    : _technique (getTechnique (iTechniqueChar)) {
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::
  PartnershipTechnique (const std::string& iTechniqueStr) {
    //
    const size_t lSize = iTechniqueStr.size();
    assert (lSize == 1);
    const char lTechniqueChar = iTechniqueStr[0];
    _technique = getTechnique (lTechniqueChar);
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& PartnershipTechnique::
  getLabel (const EN_PartnershipTechnique& iTechnique) {
    return _labels[iTechnique];
  }

  // ////////////////////////////////////////////////////////////////////
  char PartnershipTechnique::
  getTechniqueLabel (const EN_PartnershipTechnique& iTechnique) {
    return _techniqueLabels[iTechnique];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PartnershipTechnique::
  getTechniqueLabelAsString (const EN_PartnershipTechnique& iTechnique) {
    std::ostringstream oStr;
    oStr << _techniqueLabels[iTechnique];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PartnershipTechnique::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx] << " (" << _techniqueLabels[idx] << ")";
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::EN_PartnershipTechnique
  PartnershipTechnique::getTechnique() const {
    return _technique;
  }
```

```

// ////////////////////////////////////////////////////////////////////
char PartnershipTechnique::getTechniqueAsChar() const {
  const char oTechniqueChar = _techniqueLabels[_technique];
  return oTechniqueChar;
}

// ////////////////////////////////////////////////////////////////////
std::string PartnershipTechnique::getTechniqueAsString() const {
  std::ostringstream oStr;
  oStr << _techniqueLabels[_technique];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string PartnershipTechnique::describe() const {
  std::ostringstream ostr;
  ostr << _labels[_technique];
  return ostr.str();
}

// ////////////////////////////////////////////////////////////////////
bool PartnershipTechnique::
operator== (const EN_PartnershipTechnique& iTechnique) const {
  return (_technique == iTechnique);
}

}
```

33.95 stdair/basic/PartnershipTechnique.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::PartnershipTechnique

  Enumeration of partnership techniques.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.96 stdair/basic/PartnershipTechnique.hpp

```
#ifndef __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
#define __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct PartnershipTechnique : public StructAbstract {
  public:
    typedef enum {
      NONE = 0,
      RAE_DA,
      RAE_YP,
      IBP_DA,
      IBP_YP,
      IBP_YP_U,
      RMC,
      A_RMC,
      LAST_VALUE
    } EN_PartnershipTechnique;

    static const std::string& getLabel (const EN_PartnershipTechnique&);

    static EN_PartnershipTechnique getTechnique (const char);

    static char getTechniqueLabel (const EN_PartnershipTechnique&);

    static std::string getTechniqueLabelAsString (const EN_PartnershipTechnique&);

    static std::string describeLabels();

    EN_PartnershipTechnique getTechnique() const;

    char getTechniqueAsChar() const;

    std::string getTechniqueAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_PartnershipTechnique&) const;

  public:
    PartnershipTechnique (const EN_PartnershipTechnique&);
    PartnershipTechnique (const char iTechnique);
    PartnershipTechnique (const std::string& iTechnique);

    PartnershipTechnique (const PartnershipTechnique&);

  private:
    PartnershipTechnique();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _techniqueLabels[LAST_VALUE];

  private:
```

```
    // //////// Attributes /////////
    EN_PartnershipTechnique _technique;
  };

}
#endif // __STDAIR_BAS_PARTNERSHIPTECHNIQUE_HPP
```

33.97 stdair/basic/PassengerChoiceModel.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerChoiceModel.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.98 stdair/basic/PassengerChoiceModel.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PassengerChoiceModel.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string PassengerChoiceModel::_labels[LAST_VALUE] =
00015     { "HardRestrictionModel", "PriceOrientedModel", "HybridModel"};
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   const char PassengerChoiceModel::
00019   _modelLabels[LAST_VALUE] = { 'R', 'P', 'H'};
00020
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   PassengerChoiceModel::PassengerChoiceModel()
00024     : _model (LAST_VALUE) {
00025     assert (false);
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   PassengerChoiceModel::
00030   PassengerChoiceModel (const PassengerChoiceModel& iPassengerChoiceModel)
00031     : _model (iPassengerChoiceModel._model) {
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   PassengerChoiceModel::
00036   PassengerChoiceModel (const EN_PassengerChoiceModel& iPassengerChoiceModel)
00037     : _model (iPassengerChoiceModel) {
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   PassengerChoiceModel::PassengerChoiceModel (const char iModel) {
00042     switch (iModel) {
00043     case 'R': _model = HARD_RESTRICTION; break;
00044     case 'P': _model = PRICE_ORIENTED; break;
00045     case 'H': _model = HYBRID; break;
00046     default: _model = LAST_VALUE; break;
00047     }
00048
00049     if (_model == LAST_VALUE) {
00050       const std::string& lLabels = describeLabels();
00051       std::ostringstream oMessage;
00052       oMessage << "The passenger choice model '"
00053                << "' is not known. Known passenger choice models " << lLabels;
00054       throw stdair::CodeConversionException (oMessage.str());
00055     }
00056   }
00057
00058   // ////////////////////////////////////////////////////////////////////
00059   const std::string& PassengerChoiceModel::
00060   getLabel (const EN_PassengerChoiceModel& iModel) {
00061     return _labels[iModel];
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   char PassengerChoiceModel::getModelLabel (const EN_PassengerChoiceModel& iModel
```

```
) {
    return _modelLabels[iModel];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PassengerChoiceModel::
  getModelLabelAsString (const EN_PassengerChoiceModel& iModel) {
    std::ostringstream oStr;
    oStr << _modelLabels[iModel];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PassengerChoiceModel::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx] << " (" << _modelLabels[idx] << ")";
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  PassengerChoiceModel::EN_PassengerChoiceModel PassengerChoiceModel::getModel()
const {
    return _model;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PassengerChoiceModel::getModelAsString() const {
    std::ostringstream oStr;
    oStr << _modelLabels[_model];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string PassengerChoiceModel::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_model];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool PassengerChoiceModel::
  operator== (const EN_PassengerChoiceModel& iModel) const {
    return (_model == iModel);
  }

}
```

33.99 stdair/basic/PassengerChoiceModel.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::PassengerChoiceModel

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.100 stdair/basic/PassengerChoiceModel.hpp

```
#ifndef __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP
#define __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct PassengerChoiceModel : public stdair::StructAbstract {
  public:
    typedef enum {
      HARD_RESTRICTION = 0,
      PRICE_ORIENTED,
      HYBRID,
      LAST_VALUE
    } EN_PassengerChoiceModel;

    static const std::string& getLabel (const EN_PassengerChoiceModel&);

    static char getModelLabel (const EN_PassengerChoiceModel&);

    static std::string getModelLabelAsString (const EN_PassengerChoiceModel&);

    static std::string describeLabels();

    EN_PassengerChoiceModel getModel() const;

    std::string getModelAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_PassengerChoiceModel&) const;

  public:
    PassengerChoiceModel (const EN_PassengerChoiceModel&);
    PassengerChoiceModel (const char iModel);
    PassengerChoiceModel (const PassengerChoiceModel&);

  private:
    PassengerChoiceModel();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _modelLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_PassengerChoiceModel _model;
  };
}

#endif // __STDAIR_BAS_PASSENGERCHOICEMODEL_HPP
```

33.101 stdair/basic/PassengerType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PassengerType.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.102 stdair/basic/PassengerType.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/stdair_exceptions.hpp>
00009 #include <stdair/basic/PassengerType.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   const std::string PassengerType::_labels[LAST_VALUE] =
00015     { "Leisure", "Business", "First" };
00016
00017   const char PassengerType::_typeLabels[LAST_VALUE] = { 'L', 'B', 'F' };
00018
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   PassengerType::PassengerType (const EN_PassengerType& iPassengerType)
00022     : _type (iPassengerType) {
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   PassengerType::PassengerType (const char iType) {
00027     switch (iType) {
00028     case 'L': _type = LEISURE; break;
00029     case 'B': _type = BUSINESS; break;
00030     case 'F': _type = FIRST; break;
00031     default: _type = LAST_VALUE; break;
00032     }
00033
00034     if (_type == LAST_VALUE) {
00035       const std::string& lLabels = describeLabels();
00036       std::ostringstream oMessage;
00037       oMessage << "The passenger type '" << iType
00038                << "' is not known. Known passenger types: " << lLabels;
00039       throw CodeConversionException (oMessage.str());
00040     }
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   const std::string& PassengerType::getLabel (const EN_PassengerType& iType) {
00045     return _labels[iType];
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   char PassengerType::getTypeLabel (const EN_PassengerType& iType) {
00050     return _typeLabels[iType];
00051   }
00052
00053   // ////////////////////////////////////////////////////////////////////
00054   std::string PassengerType::
00055   getTypeLabelAsString (const EN_PassengerType& iType) {
00056     std::ostringstream oStr;
00057     oStr << _typeLabels[iType];
00058     return oStr.str();
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   std::string PassengerType::describeLabels() {
00063     std::ostringstream ostr;
00064     for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
00065       if (idx != 0) {
```

```
        ostr << ", ";
      }
      ostr << _labels[idx];
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  PassengerType::EN_PassengerType PassengerType::getType() const {
    return _type;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PassengerType::getTypeAsString() const {
    std::ostringstream oStr;
    oStr << _typeLabels[_type];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string PassengerType::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_type];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool PassengerType::operator== (const EN_PassengerType& iType) const {
    return (_type == iType);
  }

}
```

33.103 stdair/basic/PassengerType.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::PassengerType

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.104 stdair/basic/PassengerType.hpp

```
#ifndef __STDAIR_BAS_PASSENGERTYPE_HPP
#define __STDAIR_BAS_PASSENGERTYPE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct PassengerType : public StructAbstract {
  public:
    typedef enum {
      LEISURE = 0,
      BUSINESS,
      FIRST,
      LAST_VALUE
    } EN_PassengerType;

    static const std::string& getLabel (const EN_PassengerType&);

    static char getTypeLabel (const EN_PassengerType&);

    static std::string getTypeLabelAsString (const EN_PassengerType&);

    static std::string describeLabels();

    EN_PassengerType getType() const;

    std::string getTypeAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_PassengerType&) const;

  public:
    PassengerType (const EN_PassengerType&);
    PassengerType (const char iType);


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _typeLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_PassengerType _type;
  };

}
#endif // __STDAIR_BAS_PASSENGERTYPE_HPP
```

33.105 stdair/basic/PreOptimisationMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.106 stdair/basic/PreOptimisationMethod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string PreOptimisationMethod::_labels[LAST_VALUE] =
    {"None", "Fare Adjustment", "Marginal Revenue Transformation"};

  // ////////////////////////////////////////////////////////////////////
  const char PreOptimisationMethod::
  _methodLabels[LAST_VALUE] = {'N', 'F', 'M'};


  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::PreOptimisationMethod()
    : _method (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::
  PreOptimisationMethod (const PreOptimisationMethod& iPreOptimisationMethod)
    : _method (iPreOptimisationMethod._method) {
  }

  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::
  PreOptimisationMethod (const EN_PreOptimisationMethod& iPreOptimisationMethod)
    : _method (iPreOptimisationMethod) {
  }

  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::PreOptimisationMethod (const char iMethod) {
    switch (iMethod) {
    case 'N': _method = NONE; break;
    case 'F': _method = FA; break;
    case 'M': _method = MRT; break;
    default: _method = LAST_VALUE; break;
    }

    if (_method == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The pre-optimisation method '" << iMethod
               << "' is not known. Known pre-optimisation methods: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& PreOptimisationMethod::
  getLabel (const EN_PreOptimisationMethod& iMethod) {
    return _labels[iMethod];
  }

  // ////////////////////////////////////////////////////////////////////
  char PreOptimisationMethod::getMethodLabel (const EN_PreOptimisationMethod& iMe
```

```
thod) {
    return _methodLabels[iMethod];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PreOptimisationMethod::
  getMethodLabelAsString (const EN_PreOptimisationMethod& iMethod) {
    std::ostringstream oStr;
    oStr << _methodLabels[iMethod];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PreOptimisationMethod::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::EN_PreOptimisationMethod
PreOptimisationMethod::getMethod() const {
    return _method;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PreOptimisationMethod::getMethodAsString() const {
    std::ostringstream oStr;
    oStr << _methodLabels[_method];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string PreOptimisationMethod::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_method];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool PreOptimisationMethod::
  operator== (const EN_PreOptimisationMethod& iMethod) const {
    return (_method == iMethod);
  }

}
```

33.107 stdair/basic/PreOptimisationMethod.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::PreOptimisationMethod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.108 stdair/basic/PreOptimisationMethod.hpp

```
#ifndef __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP
#define __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct PreOptimisationMethod : public StructAbstract {
  public:
    typedef enum {
      NONE = 0,
      FA,
      MRT,
      LAST_VALUE
    } EN_PreOptimisationMethod;

    static const std::string& getLabel (const EN_PreOptimisationMethod&);

    static char getMethodLabel (const EN_PreOptimisationMethod&);

    static std::string getMethodLabelAsString (const EN_PreOptimisationMethod&);

    static std::string describeLabels();

    EN_PreOptimisationMethod getMethod() const;

    std::string getMethodAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_PreOptimisationMethod&) const;

  public:
    PreOptimisationMethod (const EN_PreOptimisationMethod&);
    PreOptimisationMethod (const char iMethod);
    PreOptimisationMethod (const PreOptimisationMethod&);

  private:
    PreOptimisationMethod();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _methodLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_PreOptimisationMethod _method;
  };

}
#endif // __STDAIR_BAS_PREOPTIMISATIONMETHOD_HPP
```

33.109 stdair/basic/ProgressStatus.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/ProgressStatus.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.110 stdair/basic/ProgressStatus.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/basic/ProgressStatus.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  ProgressStatus::ProgressStatus (const Count_T& iCurrentNb,
                                  const Count_T& iExpectedNb,
                                  const Count_T& iActualNb)
    : _currentNb (iCurrentNb),
      _expectedNb (iExpectedNb), _actualNb (iActualNb) {
  }

  // ////////////////////////////////////////////////////////////////////
  ProgressStatus::ProgressStatus (const Count_T& iExpectedNb,
                                  const Count_T& iActualNb)
    : _currentNb (DEFAULT_PROGRESS_STATUS),
      _expectedNb (iExpectedNb), _actualNb (iActualNb) {
  }

  // ////////////////////////////////////////////////////////////////////
  ProgressStatus::ProgressStatus (const Count_T& iExpectedNb)
    : _currentNb (DEFAULT_PROGRESS_STATUS),
      _expectedNb (iExpectedNb), _actualNb (iExpectedNb) {
  }

  // ////////////////////////////////////////////////////////////////////
  ProgressStatus::ProgressStatus()
    : _currentNb (DEFAULT_PROGRESS_STATUS),
      _expectedNb (DEFAULT_PROGRESS_STATUS),
      _actualNb (DEFAULT_PROGRESS_STATUS) {
  }

  // ////////////////////////////////////////////////////////////////////
  ProgressStatus::ProgressStatus (const ProgressStatus& iProgressStatus)
    : _currentNb (iProgressStatus._currentNb),
      _expectedNb (iProgressStatus._expectedNb),
      _actualNb (iProgressStatus._actualNb) {
  }

  // ////////////////////////////////////////////////////////////////////
  void ProgressStatus::reset() {
    _currentNb = DEFAULT_PROGRESS_STATUS;
    _actualNb = DEFAULT_PROGRESS_STATUS;
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string ProgressStatus::describe() const {
    std::ostringstream oStr;
    oStr << _currentNb << " / {" << _expectedNb << ", " << _actualNb << "}";
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string ProgressStatus::toString() const {
    std::ostringstream oStr;
    oStr << std::setprecision (3) << progress()
```

```
00066          << "%  (" << _currentNb << "/" << _actualNb << ")";
00067     return oStr.str();
00068   }
00069
00070 }
```

33.111 stdair/basic/ProgressStatus.hpp File Reference

```
#include <string>
#include <boost/progress.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/service/Logger.hpp>
```

Classes

- struct stdair::ProgressStatus

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.112 stdair/basic/ProgressStatus.hpp

```
#ifndef __STDAIR_BAS_PROGRESSSTATUS_HPP
#define __STDAIR_BAS_PROGRESSSTATUS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost Progress
#include <boost/progress.hpp>
// StdAir
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  struct ProgressStatus : public StructAbstract {
  public:
    // ///////////////// Getters /////////////////
    const Count_T& count() const {
      return _currentNb;
    }

    const Count_T& getCurrentNb() const {
      return _currentNb;
    }

    const Count_T& getExpectedNb() const {
      return _expectedNb;
    }

    const Count_T& getActualNb() const {
      return _actualNb;
    }

    const ProgressPercentage_T progress() const {
      if (_actualNb == 0) {
        return 0;
      }
      Percentage_T lPercentage =
        (static_cast<Percentage_T> (_currentNb)
         / static_cast<Percentage_T> (_actualNb));
      lPercentage *= MAXIMUM_PROGRESS_STATUS;
      return lPercentage;
    }


    // ///////////////// Setters /////////////////
    void setCurrentNb (const Count_T& iCurrentNb) {
      _currentNb = iCurrentNb;
    }

    void setExpectedNb (const Count_T& iExpectedNb) {
      _expectedNb = iExpectedNb;
    }

    void setActualNb (const Count_T& iActualNb) {
      _actualNb = iActualNb;
    }

    void reset();

    Count_T operator+= (Count_T iIncrement) {
```

```
00084       _currentNb += iIncrement;
00085       return _currentNb;
00086     }
00087
00089     Count_T operator++() {
00090       ++_currentNb;
00091       return _currentNb;
00092     }
00093
00094
00095   public:
00096     // ///////////////// Display Support Methods /////////////////
00098     const std::string describe() const;
00099
00101     const std::string toString() const;
00102
00103
00104   public:
00112     ProgressStatus (const Count_T& iCurrentNb, const Count_T& iExpectedNb,
00113                     const Count_T& iActualNb);
00114
00123     ProgressStatus (const Count_T& iExpectedNb, const Count_T& iActualNb);
00124
00133     ProgressStatus (const Count_T& iActualNb);
00134
00140     ProgressStatus();
00141
00145     ProgressStatus (const ProgressStatus&);
00146
00147   private:
00148     // ///////////////// Attributes /////////////////
00150     Count_T _currentNb;
00151
00153     Count_T _expectedNb;
00154
00156     Count_T _actualNb;
00157   };
00158
00159 }
00160 #endif // __STDAIR_BAS_PROGRESSSTATUS_HPP
```

33.113 stdair/basic/ProgressStatusSet.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/ProgressStatusSet.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.114 stdair/basic/ProgressStatusSet.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/ProgressStatusSet.hpp>
00009
00010 namespace stdair {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   ProgressStatusSet::ProgressStatusSet()
00014     : _eventType (EventType::LAST_VALUE), _typeSpecificProgressStatus(),
00015       _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00016     assert (false);
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   ProgressStatusSet::ProgressStatusSet (const EventType::EN_EventType& iType)
00021     : _eventType (iType), _typeSpecificProgressStatus(),
00022       _generatorProgressStatus(), _overallProgressStatus(), _generatorKey ("") {
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   ProgressStatusSet::
00027     ProgressStatusSet (const ProgressStatusSet& iProgressStatusSet)
00028       : _eventType (iProgressStatusSet._eventType),
00029         _typeSpecificProgressStatus(iProgressStatusSet._typeSpecificProgressStatu
      s),
00030         _generatorProgressStatus (iProgressStatusSet._generatorProgressStatus),
00031         _overallProgressStatus (iProgressStatusSet._overallProgressStatus),
00032         _generatorKey (iProgressStatusSet._generatorKey) {
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   ProgressStatusSet::~ProgressStatusSet() {
00037   }
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   void ProgressStatusSet::fromStream (std::istream& ioIn) {
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   const std::string ProgressStatusSet::describe() const {
00045     std::ostringstream oStr;
00046
00047     oStr << "-[Overall]"
00048          << "[" << _overallProgressStatus.getCurrentNb()
00049          << "/{" << _overallProgressStatus.getExpectedNb()
00050          << "," << _overallProgressStatus.getActualNb()
00051          << "}] ";
00052
00053     oStr << "[" << EventType (_eventType) << "]"
00054          << "[" << _typeSpecificProgressStatus.getCurrentNb()
00055          << "/{" << _typeSpecificProgressStatus.getExpectedNb()
00056          << "," << _typeSpecificProgressStatus.getActualNb()
00057          << "}]";
00058
00059     oStr << " [Specific generator: " << _generatorKey << "]"
00060          << "[" << _generatorProgressStatus.getCurrentNb()
00061          << "/{" << _generatorProgressStatus.getExpectedNb()
00062          << "," << _generatorProgressStatus.getActualNb()
00063          << "}]";
00064
```

```
    return oStr.str();
  }

}
```

33.115 stdair/basic/ProgressStatusSet.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/basic/ProgressStatus.hpp>
```

Classes

- struct stdair::ProgressStatusSet

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.116 stdair/basic/ProgressStatusSet.hpp

```
00001 #ifndef __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00002 #define __STDAIR_BAS_PROGRESSSTATUSSET_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_event_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/basic/EventType.hpp>
00015 #include <stdair/basic/ProgressStatus.hpp>
00016
00017 namespace stdair {
00018
00022   struct ProgressStatusSet : public StructAbstract {
00023     // ////////////// Getters ///////////
00031     const ProgressStatus& getTypeSpecificStatus() const {
00032       return _typeSpecificProgressStatus;
00033     }
00034
00043     const ProgressStatus& getSpecificGeneratorStatus() const {
00044       return _generatorProgressStatus;
00045     }
00046
00054     const ProgressStatus& getOverallStatus() const {
00055       return _overallProgressStatus;
00056     }
00057
00058
00059     // ////////////// Setters ///////////
00060   public:
00062     void setTypeSpecificStatus (const ProgressStatus& iProgressStatus) {
00063       _typeSpecificProgressStatus = iProgressStatus;
00064     }
00065
00068     void setSpecificGeneratorStatus (const ProgressStatus& iProgressStatus,
00069                                      const EventGeneratorKey_T& iKey) {
00070       _generatorProgressStatus = iProgressStatus;
00071       _generatorKey = iKey;
00072     }
00073
00076     void setOverallStatus (const ProgressStatus& iProgressStatus) {
00077       _overallProgressStatus = iProgressStatus;
00078     }
00079
00080
00081     // ////////// Display methods //////////
00082   public:
00085     void fromStream (std::istream& ioIn);
00086
00088     const std::string describe() const;
00089
00090
00091     // ////////// Constructors and destructors /////////
00092   public:
00094     ProgressStatusSet (const EventType::EN_EventType&);
00096     ProgressStatusSet (const ProgressStatusSet&);
00098     ~ProgressStatusSet();
00099
00100   private:
00102     ProgressStatusSet ();
```

```
00103
00104     // ////////////////// Attributes ////////////////////
00105   private:
00109     const EventType::EN_EventType _eventType;
00110
00114     ProgressStatus _typeSpecificProgressStatus;
00115
00119     ProgressStatus _generatorProgressStatus;
00120
00124     ProgressStatus _overallProgressStatus;
00125
00129     EventGeneratorKey_T _generatorKey;
00130   };
00131
00132 }
00133 #endif // __STDAIR_BAS_PROGRESSSTATUSSET_HPP
```

33.117 stdair/basic/RandomGeneration.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/version.hpp>
#include <stdair/basic/RandomGeneration.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.118 stdair/basic/RandomGeneration.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/version.hpp>
00009 #if BOOST_VERSION >= 103500
00010 #include <boost/math/distributions/normal.hpp>
00011 #endif // BOOST_VERSION >= 103500
00012 // StdAir
00013 #include <stdair/basic/RandomGeneration.hpp>
00014
00015 namespace stdair {
00016
00022   // ////////////////////////////////////////////////////////////////////
00023   RandomGeneration::RandomGeneration() : _generator (1) {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   RandomGeneration::RandomGeneration (const RandomSeed_T& iSeed)
00028     : _generator (iSeed) {
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   RandomGeneration::RandomGeneration (const RandomGeneration& iRandomGeneration)
00033     : _generator (iRandomGeneration._generator) {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   RandomGeneration::~RandomGeneration() {
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   void RandomGeneration::init (const RandomSeed_T& iSeed) {
00042     _generator.seed (iSeed);
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string RandomGeneration::describe() const {
00047     std::ostringstream oStr;
00048     oStr << _generator;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   RealNumber_T RandomGeneration::generateUniform01() {
00054     UniformGenerator_T lGenerator (_generator, boost::uniform_real<>(0, 1));
00055     return lGenerator();
00056   }
00057
00058   // ////////////////////////////////////////////////////////////////////
00059   RealNumber_T RandomGeneration::generateUniform(const RealNumber_T& iMinValue,
00060                                                  const RealNumber_T& iMaxValue) {

00061     const Probability_T lVariateUnif01 = generateUniform01();
00062     const RealNumber_T lVariateUnif =
00063       iMinValue + lVariateUnif01 * (iMaxValue - iMinValue);
00064     return lVariateUnif;
00065   }
00066
00067   // ////////////////////////////////////////////////////////////////////
00068   RealNumber_T RandomGeneration::generateNormal (const RealNumber_T& mu,
00069                                                  const RealNumber_T& sigma) {
```

```

#if BOOST_VERSION >= 103500
    const Probability_T lVariateUnif = generateUniform01();
    const boost::math::normal lNormal (mu, sigma);
    const RealNumber_T lRealNumberOfRequestsToBeGenerated =
      boost::math::quantile (lNormal, lVariateUnif);
#else // BOOST_VERSION >= 103500
    // TODO: rely on GSL when Boost version smaller than 1.35
    const RealNumber_T lRealNumberOfRequestsToBeGenerated = 0.0;
#endif // BOOST_VERSION >= 103500

    return lRealNumberOfRequestsToBeGenerated;

  }

  // ////////////////////////////////////////////////////////////////////
  RealNumber_T RandomGeneration::generateExponential (const RealNumber_T& lambda)
  {
    ExponentialDistribution_T lExponentialDistribution (lambda);

    ExponentialGenerator_T lExponentialDistributionGenerator (_generator,
                                                              lExponentialDistribution);

    // Generate a random variate, expressed in (fractional) day
    const RealNumber_T lExponentialVariateInDays =
      lExponentialDistributionGenerator();

    return lExponentialVariateInDays;
  }

}
```

33.119 stdair/basic/RandomGeneration.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::RandomGeneration

  Class holding a random generator.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.120 stdair/basic/RandomGeneration.hpp

```
00001 #ifndef __STDAIR_BAS_RANDOMGENERATION_HPP
00002 #define __STDAIR_BAS_RANDOMGENERATION_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/stdair_maths_types.hpp>
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00017   struct RandomGeneration : public StructAbstract {
00018   public:
00019     // ///////////// Business Methods /////////////
00024     RealNumber_T generateUniform01();
00025
00030     RealNumber_T operator()() {
00031       return generateUniform01();
00032     }
00033
00039     RealNumber_T generateUniform (const RealNumber_T&, const RealNumber_T&);
00040
00045     RealNumber_T generateNormal (const RealNumber_T&, const RealNumber_T&);
00046
00051     RealNumber_T generateExponential (const RealNumber_T&);
00052
00056     BaseGenerator_T& getBaseGenerator () { return _generator; }
00057
00058
00059   public:
00060     // ////////////// Display Support Methods //////////
00064     const std::string describe() const;
00065
00066
00067   public:
00068     // ////////// Constructors and destructors /////////
00072     RandomGeneration (const RandomSeed_T&);
00076     RandomGeneration();
00077
00078   private:
00082     RandomGeneration (const RandomGeneration&);
00086     RandomGeneration& operator= (const RandomGeneration& iRandomGeneration) {
00087       _generator = iRandomGeneration._generator;
00088       return *this;
00089     }
00090   public:
00094     ~RandomGeneration();
00095
00103     void init (const RandomSeed_T&);
00104
00105     // ////////// Attributes //////////
00112     BaseGenerator_T _generator;
00113   };
00114
00115 }
00116 #endif // __STDAIR_BAS_RANDOMGENERATION_HPP
```

33.121 stdair/basic/SampleType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/SampleType.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.122 stdair/basic/SampleType.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/SampleType.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string SampleType::_labels[LAST_VALUE] =
    { "All", "AllForPartnerships", "RevenueManagement", "Inventory", "Schedule",
      "RevenueAccounting", "FareQuote", "CRS", "DemandGeneration", "EventManageme
nt",
      "CustomerChoice" };

  // ////////////////////////////////////////////////////////////////////
  const char SampleType::
  _typeLabels[LAST_VALUE] = { 'A', 'P', 'R', 'I', 'S', 'T', 'F', 'C', 'D', 'E', '
M' };


  // ////////////////////////////////////////////////////////////////////
  SampleType::SampleType()
    : _type (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SampleType::SampleType (const SampleType& iSampleType)
    : _type (iSampleType._type) {
  }

  // ////////////////////////////////////////////////////////////////////
  SampleType::SampleType (const EN_SampleType& iSampleType)
    : _type (iSampleType) {
  }

  // ////////////////////////////////////////////////////////////////////
  SampleType::SampleType (const char iType) {
    switch (iType) {
    case 'A': _type = ALL; break;
    case 'P': _type = A4P; break;
    case 'R': _type = RMS; break;
    case 'I': _type = INV; break;
    case 'S': _type = SCH; break;
    case 'T': _type = RAC; break;
    case 'F': _type = FQT; break;
    case 'C': _type = CRS; break;
    case 'D': _type = DEM; break;
    case 'E': _type = EVT; break;
    case 'M': _type = CCM; break;
    default: _type = LAST_VALUE; break;
    }

    if (_type == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The sample type '" << iType
               << "' is not known. Known sample types: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& SampleType::getLabel (const EN_SampleType& iType) {
    return _labels[iType];
  }

  // ////////////////////////////////////////////////////////////////////
  char SampleType::getTypeLabel (const EN_SampleType& iType) {
    return _typeLabels[iType];
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SampleType::getTypeLabelAsString (const EN_SampleType& iType) {
    std::ostringstream oStr;
    oStr << _typeLabels[iType];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SampleType::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx];
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  SampleType::EN_SampleType SampleType::getType() const {
    return _type;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SampleType::getTypeAsString() const {
    std::ostringstream oStr;
    oStr << _typeLabels[_type];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SampleType::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_type];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool SampleType::operator== (const EN_SampleType& iType) const {
    return (_type == iType);
  }

}
```

33.123 stdair/basic/SampleType.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::SampleType

  Enumeration of BOM sample types.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.124 stdair/basic/SampleType.hpp

```
#ifndef __STDAIR_BAS_SAMPLETYPE_HPP
#define __STDAIR_BAS_SAMPLETYPE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct SampleType : public StructAbstract {
  public:
    typedef enum {
      ALL = 0,
      A4P,
      RMS,
      INV,
      SCH,
      RAC,
      FQT,
      CRS,
      DEM,
      EVT,
      CCM,
      LAST_VALUE
    } EN_SampleType;

    static const std::string& getLabel (const EN_SampleType&);

    static char getTypeLabel (const EN_SampleType&);

    static std::string getTypeLabelAsString (const EN_SampleType&);

    static std::string describeLabels();

    EN_SampleType getType() const;

    std::string getTypeAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_SampleType&) const;

  public:
    SampleType (const EN_SampleType&);
    SampleType (const char iType);
    SampleType (const SampleType&);

  private:
    SampleType();


  private:
    static const std::string _labels[LAST_VALUE];

    static const char _typeLabels[LAST_VALUE];


  private:
    // //////// Attributes /////////
    EN_SampleType _type;
```

```
00122   };
00123
00124 }
00125 #endif // __STDAIR_BAS_SAMPLETYPE_HPP
```

33.125 stdair/basic/ServiceInitialisationType.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.126 stdair/basic/ServiceInitialisationType.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string ServiceInitialisationType::_labels[LAST_VALUE] =
    { "Not yet initialised", "File parsing",  "Built-in sample BOM" };

  // ////////////////////////////////////////////////////////////////////
  const char ServiceInitialisationType::_typeLabels[LAST_VALUE] =
    { 'N', 'F', 'B' };


  // ////////////////////////////////////////////////////////////////////
  ServiceInitialisationType::ServiceInitialisationType()
    : _type (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  ServiceInitialisationType::
  ServiceInitialisationType (const ServiceInitialisationType& iServiceInitialisationType)
    : _type (iServiceInitialisationType._type) {
  }

  // ////////////////////////////////////////////////////////////////////
  ServiceInitialisationType::
  ServiceInitialisationType (const EN_ServiceInitialisationType& iServiceInitialisationType)
    : _type (iServiceInitialisationType) {
  }

  // ////////////////////////////////////////////////////////////////////
  ServiceInitialisationType::EN_ServiceInitialisationType
  ServiceInitialisationType::getType (const char iTypeChar) {
    EN_ServiceInitialisationType oType;
    switch (iTypeChar) {
    case 'N': oType = NOT_YET_INITIALISED; break;
    case 'F': oType = FILE_PARSING; break;
    case 'B': oType = BUILTIN_SAMPLE; break;
    default: oType = LAST_VALUE; break;
    }

    if (oType == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The service initialisation type '" << iTypeChar
               << "' is not known. "
               << "Known service initialisation types: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }

    return oType;
  }

  // ////////////////////////////////////////////////////////////////////
```

```
ServiceInitialisationType::
ServiceInitialisationType (const char iTypeChar)
  : _type (getType (iTypeChar)) {
}

// ////////////////////////////////////////////////////////////////////
ServiceInitialisationType::
ServiceInitialisationType (const std::string& iTypeStr) {
  //
  const size_t lSize = iTypeStr.size();
  assert (lSize == 1);
  const char lTypeChar = iTypeStr[0];
  _type = getType (lTypeChar);
}

// ////////////////////////////////////////////////////////////////////
const std::string& ServiceInitialisationType::
getLabel (const EN_ServiceInitialisationType& iType) {
  return _labels[iType];
}

// ////////////////////////////////////////////////////////////////////
char ServiceInitialisationType::
getTypeLabel (const EN_ServiceInitialisationType& iType) {
  return _typeLabels[iType];
}

// ////////////////////////////////////////////////////////////////////
std::string ServiceInitialisationType::
getTypeLabelAsString (const EN_ServiceInitialisationType& iType) {
  std::ostringstream oStr;
  oStr << _typeLabels[iType];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
std::string ServiceInitialisationType::describeLabels() {
  std::ostringstream ostr;
  for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
    if (idx != 0) {
      ostr << ", ";
    }
    ostr << _labels[idx];
  }
  return ostr.str();
}

// ////////////////////////////////////////////////////////////////////
ServiceInitialisationType::EN_ServiceInitialisationType
ServiceInitialisationType::getType() const {
  return _type;
}

// ////////////////////////////////////////////////////////////////////
char ServiceInitialisationType::getTypeAsChar() const {
  const char oTypeChar = _typeLabels[_type];
  return oTypeChar;
}

// ////////////////////////////////////////////////////////////////////
std::string ServiceInitialisationType::getTypeAsString() const {
  std::ostringstream oStr;
  oStr << _typeLabels[_type];
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
```

```
00131   const std::string ServiceInitialisationType::describe() const {
00132     std::ostringstream ostr;
00133     ostr << _labels[_type];
00134     return ostr.str();
00135   }
00136
00137   // ////////////////////////////////////////////////////////////////////
00138   bool ServiceInitialisationType::
00139   operator== (const EN_ServiceInitialisationType& iType) const {
00140     return (_type == iType);
00141   }
00142
00143 }
```

33.127 stdair/basic/ServiceInitialisationType.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::ServiceInitialisationType

  Enumeration of service initialisation types.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.128 stdair/basic/ServiceInitialisationType.hpp

```
#ifndef __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP
#define __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  struct ServiceInitialisationType : public StructAbstract {
  public:
    typedef enum {
      NOT_YET_INITIALISED = 0,
      FILE_PARSING,
      BUILTIN_SAMPLE,
      LAST_VALUE
    } EN_ServiceInitialisationType;

    static const std::string& getLabel (const EN_ServiceInitialisationType&);

    static EN_ServiceInitialisationType getType (const char);

    static char getTypeLabel (const EN_ServiceInitialisationType&);

    static std::string
    getTypeLabelAsString (const EN_ServiceInitialisationType&);

    static std::string describeLabels();

    EN_ServiceInitialisationType getType() const;

    char getTypeAsChar() const;

    std::string getTypeAsString() const;

    const std::string describe() const;

  public:
    bool operator== (const EN_ServiceInitialisationType&) const;

  public:
    ServiceInitialisationType (const EN_ServiceInitialisationType&);
    ServiceInitialisationType (const char iType);
    ServiceInitialisationType (const std::string& iType);
    ServiceInitialisationType (const ServiceInitialisationType&);

  private:
    ServiceInitialisationType();


  private:
    static const std::string _labels[LAST_VALUE];
    static const char _typeLabels[LAST_VALUE];

  private:
    // //////// Attributes /////////
    EN_ServiceInitialisationType _type;
  };

}
#endif // __STDAIR_BAS_SERVICEINITIALISATIONTYPE_HPP
```

33.129 stdair/basic/StructAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- struct stdair::StructAbstract

  Base class for the light structures.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template<class charT , class traits >
  std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::StructAbstract &iStruct)
- template<class charT , class traits >
  std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::StructAbstract &ioStruct)

33.129.1 Function Documentation

33.129.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::StructAbstract & *iStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 61 of file StructAbstract.hpp.

33.129.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::StructAbstract & *ioStruct*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 89 of file StructAbstract.hpp.

References stdair::StructAbstract::fromStream().

33.130 stdair/basic/StructAbstract.hpp

```
#ifndef __STDAIR_BAS_STRUCTABSTRACT_HPP
#define __STDAIR_BAS_STRUCTABSTRACT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>

namespace stdair {

  struct StructAbstract {
  public:

    virtual ~StructAbstract() {}

    void toStream (std::ostream& ioOut) const {
      ioOut << describe();
    }

    virtual void fromStream (std::istream& ioIn) {}

    virtual const std::string describe() const = 0;

  protected:
    StructAbstract() {}
  };
}

template <class charT, class traits>
inline
std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream<charT, traits>& ioOut,
            const stdair::StructAbstract& iStruct) {
  std::basic_ostringstream<charT,traits> ostr;
  ostr.copyfmt (ioOut);
  ostr.width (0);

  // Fill string stream
  iStruct.toStream (ostr);

  // Print string stream
  ioOut << ostr.str();

  return ioOut;
}

template <class charT, class traits>
inline
std::basic_istream<charT, traits>&
operator>> (std::basic_istream<charT, traits>& ioIn,
            stdair::StructAbstract& ioStruct) {
  // Fill the Structure object with the input stream.
  ioStruct.fromStream (ioIn);
  return ioIn;

}
#endif // __STDAIR_BAS_STRUCTABSTRACT_HPP
```

33.131 stdair/basic/UnconstrainingMethod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.132 stdair/basic/UnconstrainingMethod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  const std::string UnconstrainingMethod::_labels[LAST_VALUE] =
    { "Expectation-Maximisation" };

  // ////////////////////////////////////////////////////////////////////
  const char UnconstrainingMethod::
  _methodLabels[LAST_VALUE] = { 'E' };


  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::UnconstrainingMethod()
    : _method (LAST_VALUE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::
  UnconstrainingMethod (const UnconstrainingMethod& iUnconstrainingMethod)
    : _method (iUnconstrainingMethod._method) {
  }

  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::
  UnconstrainingMethod (const EN_UnconstrainingMethod& iUnconstrainingMethod)
    : _method (iUnconstrainingMethod) {
  }

  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::UnconstrainingMethod (const char iMethod) {
    switch (iMethod) {
    case 'E': _method = EM; break;
    default: _method = LAST_VALUE; break;
    }

    if (_method == LAST_VALUE) {
      const std::string& lLabels = describeLabels();
      std::ostringstream oMessage;
      oMessage << "The unconstraining method '" << iMethod
               << "' is not known. Known unconstraining methods: " << lLabels;
      throw CodeConversionException (oMessage.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string& UnconstrainingMethod::
  getLabel (const EN_UnconstrainingMethod& iMethod) {
    return _labels[iMethod];
  }

  // ////////////////////////////////////////////////////////////////////
  char UnconstrainingMethod::getMethodLabel (const EN_UnconstrainingMethod& iMethod) {
    return _methodLabels[iMethod];
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  std::string UnconstrainingMethod::
  getMethodLabelAsString (const EN_UnconstrainingMethod& iMethod) {
    std::ostringstream oStr;
    oStr << _methodLabels[iMethod];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string UnconstrainingMethod::describeLabels() {
    std::ostringstream ostr;
    for (unsigned short idx = 0; idx != LAST_VALUE; ++idx) {
      if (idx != 0) {
        ostr << ", ";
      }
      ostr << _labels[idx] << " (" << _methodLabels[idx] << ")";
    }
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::EN_UnconstrainingMethod UnconstrainingMethod::getMethod()
    const {
    return _method;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string UnconstrainingMethod::getMethodAsString() const {
    std::ostringstream oStr;
    oStr << _methodLabels[_method];
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string UnconstrainingMethod::describe() const {
    std::ostringstream ostr;
    ostr << _labels[_method];
    return ostr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool UnconstrainingMethod::
  operator== (const EN_UnconstrainingMethod& iMethod) const {
    return (_method == iMethod);
  }

}
```

33.133 stdair/basic/UnconstrainingMethod.hpp File Reference

```
#include <string>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::UnconstrainingMethod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.134 stdair/basic/UnconstrainingMethod.hpp

```
00001 #ifndef __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
00002 #define __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/basic/StructAbstract.hpp>
00011
00012 namespace stdair {
00013
00015   struct UnconstrainingMethod : public StructAbstract {
00016   public:
00017     typedef enum {
00018       EM = 0,
00019       LAST_VALUE
00020     } EN_UnconstrainingMethod;
00021
00023     static const std::string& getLabel (const EN_UnconstrainingMethod&);
00024
00026     static char getMethodLabel (const EN_UnconstrainingMethod&);
00027
00029     static std::string getMethodLabelAsString (const EN_UnconstrainingMethod&);
00030
00032     static std::string describeLabels();
00033
00035     EN_UnconstrainingMethod getMethod() const;
00036
00038     std::string getMethodAsString() const;
00039
00042     const std::string describe() const;
00043
00044   public:
00046     bool operator== (const EN_UnconstrainingMethod&) const;
00047
00048   public:
00050     UnconstrainingMethod (const EN_UnconstrainingMethod&);
00052     UnconstrainingMethod (const char iMethod);
00054     UnconstrainingMethod (const UnconstrainingMethod&);
00055
00056   private:
00058     UnconstrainingMethod();
00059
00060
00061   private:
00063     static const std::string _labels[LAST_VALUE];
00065     static const char _methodLabels[LAST_VALUE];
00066
00067
00068   private:
00069     // //////// Attributes /////////
00071     EN_UnconstrainingMethod _method;
00072   };
00073
00074 }
00075 #endif // __STDAIR_BAS_UNCONSTRAININGMETHOD_HPP
```

33.135 stdair/basic/YieldRange.cpp File Reference

```
#include <limits>
#include <sstream>
#include <stdair/basic/YieldRange.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.136 stdair/basic/YieldRange.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <limits>
#include <sstream>
// StdAir
#include <stdair/basic/YieldRange.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  YieldRange::YieldRange() :
    _upperYield (std::numeric_limits<Yield_T>::max()),
    _averageYield (std::numeric_limits<Yield_T>::max()),
    _lowerYield (std::numeric_limits<Yield_T>::min()) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldRange::YieldRange (const YieldRange& iYieldRange) :
    _upperYield (iYieldRange.getUpperYield()),
    _averageYield (iYieldRange.getAverageYield()),
    _lowerYield (std::numeric_limits<Yield_T>::min()) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldRange::YieldRange (const Yield_T iUpperYield) :
    _upperYield (iUpperYield), _averageYield (iUpperYield),
    _lowerYield (iUpperYield) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldRange::YieldRange (const Yield_T iUpperYield,
                          const Yield_T iAverageYield) :
    _upperYield (iUpperYield), _averageYield (iAverageYield),
    _lowerYield (std::numeric_limits<Yield_T>::min()) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldRange::YieldRange (const Yield_T iUpperYield,
                          const Yield_T iAverageYield,
                          const Yield_T iLowerYield) :
    _upperYield (iUpperYield), _averageYield (iAverageYield),
    _lowerYield (iLowerYield) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldRange::~YieldRange() {
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldRange::toStream (std::ostream& ioOut) const {
    ioOut << _averageYield << "([" << _lowerYield << ", "
          << _upperYield << "])";
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldRange::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string YieldRange::describe() const {
    std::ostringstream oStr;

    return oStr.str();
```

```
00066   }
00067
00068 }
```

33.137 stdair/basic/YieldRange.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- class stdair::YieldRange

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.138 stdair/basic/YieldRange.hpp

```
#ifndef __STDAIR_BAS_YIELDRANGE_HPP
#define __STDAIR_BAS_YIELDRANGE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {

  class YieldRange : public StructAbstract {
  public:
    YieldRange ();
    YieldRange (const YieldRange&);
    YieldRange (const Yield_T iUpperYield);
    YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield);
    YieldRange (const Yield_T iUpperYield, const Yield_T iAverageYield,
                const Yield_T iLowerYield);

    virtual ~YieldRange();


    // /////////// Getters ////////////
    Yield_T getUpperYield() const {
      return _upperYield;
    }
    Yield_T getAverageYield() const {
      return _averageYield;
    }
    Yield_T getLowerYield() const {
      return _lowerYield;
    }

    // //////// Setters /////////
    void setUpperYield (const Yield_T iUpperYield) {
      _upperYield = iUpperYield;
    }
    void setAverageYield (const Yield_T iAverageYield) {
      _averageYield = iAverageYield;
    }
    void setLowerYield (const Yield_T iLowerYield) {
      _lowerYield = iLowerYield;
    }


    // ///////// Display methods ////////
    void toStream (std::ostream&) const;

    void fromStream (std::istream&);

    const std::string describe() const;

  private:
    // /////////// Attributes /////////
    Yield_T _upperYield;

    Yield_T _averageYield;

    Yield_T _lowerYield;
  };
}
#endif // __STDAIR_BAS_YIELDRANGE_HPP
```

33.139 stdair/bom/AirlineClassList.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirlineClassList.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.140 stdair/bom/AirlineClassList.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/service/Logger.hpp>
00014 #include <stdair/bom/AirlineClassList.hpp>
00015
00016 namespace stdair {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   AirlineClassList::AirlineClassList()
00020     : _key (DEFAULT_AIRLINE_CODE_LIST, DEFAULT_CLASS_CODE_LIST),
00021       _parent (NULL)  {
00022     assert (false);
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   AirlineClassList::AirlineClassList (const AirlineClassList& iACL)
00027     : _key (iACL._key),
00028       _parent (NULL),
00029       _yield(iACL._yield),
00030       _fare(iACL._fare) {
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AirlineClassList::AirlineClassList (const Key_T& iKey)
00035     : _key (iKey), _parent (NULL)  {
00036   }
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   AirlineClassList::~AirlineClassList() {
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   std::string AirlineClassList::toString() const {
00044     std::ostringstream oStr;
00045     oStr << describeKey() << ", " << _yield <<  ", " << _fare;
00046     return oStr.str();
00047   }
00048
00049   // ////////////////////////////////////////////////////////////////////
00050   void AirlineClassList::serialisationImplementationExport() const {
00051     std::ostringstream oStr;
00052     boost::archive::text_oarchive oa (oStr);
00053     oa << *this;
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   void AirlineClassList::serialisationImplementationImport() {
00058     std::istringstream iStr;
00059     boost::archive::text_iarchive ia (iStr);
00060     ia >> *this;
00061   }
00062
00063   // ////////////////////////////////////////////////////////////////////
00064   template<class Archive>
00065   void AirlineClassList::serialize (Archive& ioArchive,
```

```
                                    const unsigned int iFileVersion) {
    ioArchive & _key & _yield & _fare;
  }

}
```

33.141 stdair/bom/AirlineClassList.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
#include <stdair/bom/AirlineClassListTypes.hpp>
```

Classes

- class stdair::AirlineClassList

  Class representing the actual attributes for a segment-features.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.142 stdair/bom/AirlineClassList.hpp

```
#ifndef __STDAIR_BOM_AIRLINECLASSLIST_HPP
#define __STDAIR_BOM_AIRLINECLASSLIST_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
#include <stdair/bom/AirlineClassListTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  class AirlineClassList : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef AirlineClassListKey Key_T;


  public:
    // ////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const AirlineCodeList_T& getAirlineCodeList() const {
      return _key.getAirlineCodeList();
    }

    const ClassList_StringList_T& getClassCodeList() const {
      return _key.getClassCodeList();
    }

    const  HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const stdair::Yield_T& getYield() const {
      return _yield;
    }

    const stdair::Fare_T& getFare() const {
      return _fare;
    }

  public:
    // /////////// Setters //////////////
```

```
00080     void setYield (const Yield_T& iYield) {
00081       _yield = iYield;
00082     }
00083
00084     void setFare (const Fare_T& iFare) {
00085       _fare = iFare;
00086     }
00087
00088   public:
00089     // /////////// Display support methods /////////
00095     void toStream (std::ostream& ioOut) const {
00096       ioOut << toString();
00097     }
00098
00104     void fromStream (std::istream& ioIn) {
00105     }
00106
00110     std::string toString() const;
00111
00115     const std::string describeKey() const {
00116       return _key.toString();
00117     }
00118
00119
00120   public:
00121     // /////////// (Boost) Serialisation support methods /////////
00125     template<class Archive>
00126     void serialize (Archive& ar, const unsigned int iFileVersion);
00127
00128   private:
00133     void serialisationImplementationExport() const;
00134     void serialisationImplementationImport();
00135
00136
00137   protected:
00138     // ////////// Constructors and destructors /////////
00142     AirlineClassList (const Key_T&);
00146     virtual ~AirlineClassList();
00147
00148   private:
00152     AirlineClassList();
00153
00157     AirlineClassList (const AirlineClassList&);
00158
00159
00160   protected:
00161     // ////////// Attributes /////////
00165     Key_T _key;
00166
00170     BomAbstract* _parent;
00171
00175     HolderMap_T _holderMap;
00176
00177     /*
00178      * Yield value.
00179      */
00180     Yield_T _yield;
00181
00182     /*
00183      * Fare value.
00184      */
00185     Fare_T _fare;
00186   };
00187
00188 }
00189 #endif // __STDAIR_BOM_AIRLINECLASSLIST_HPP
00190
```

33.143 stdair/bom/AirlineClassListKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::AirlineClassListKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::AirlineClassListKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.144 stdair/bom/AirlineClassListKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/AirlineClassListKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  AirlineClassListKey::AirlineClassListKey() {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineClassListKey::
  AirlineClassListKey (const AirlineCodeList_T& iAirlineCodeList,
                       const ClassList_StringList_T& iClassCodeList)
    : _airlineCodeList (iAirlineCodeList), _classCodeList (iClassCodeList) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineClassListKey::AirlineClassListKey (const AirlineClassListKey& iKey)
    : _airlineCodeList (iKey._airlineCodeList),
      _classCodeList (iKey._classCodeList) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineClassListKey::~AirlineClassListKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineClassListKey::toStream (std::ostream& ioOut) const {
    ioOut << "AirlineClassListKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineClassListKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string AirlineClassListKey::toString() const {
    std::ostringstream oStr;
    assert (_airlineCodeList.size() == _classCodeList.size());

    unsigned short idx = 0;
    AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();
    for (ClassList_StringList_T::const_iterator itClassCode =
           _classCodeList.begin(); itClassCode != _classCodeList.end();
         ++itClassCode, ++itAirlineCode, ++idx) {
      if (idx != 0) {
        oStr << DEFAULT_KEY_SUB_FLD_DELIMITER << " ";
      }

      const AirlineCode_T& lAirlineCode = *itAirlineCode;
      const ClassCode_T& lClassCode = *itClassCode;
      oStr << lAirlineCode << " " << lClassCode;
    }
```

```

  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
void AirlineClassListKey::serialisationImplementationExport() const {
  std::ostringstream oStr;
  boost::archive::text_oarchive oa (oStr);
  oa << *this;
}

// ////////////////////////////////////////////////////////////////////
void AirlineClassListKey::serialisationImplementationImport() {
  std::istringstream iStr;
  boost::archive::text_iarchive ia (iStr);
  ia >> *this;
}

// ////////////////////////////////////////////////////////////////////
template<class Archive>
void AirlineClassListKey::serialize (Archive& ioArchive,
                                     const unsigned int iFileVersion) {
  AirlineCodeList_T::const_iterator itAirlineCode = _airlineCodeList.begin();
  for (ClassList_StringList_T::const_iterator itClassCode =
         _classCodeList.begin(); itClassCode != _classCodeList.end();
       ++itClassCode, ++itAirlineCode) {
    AirlineCode_T lAirlineCode = *itAirlineCode;
    ClassCode_T lClassCode = *itClassCode;

    ioArchive & lAirlineCode & lClassCode;
  }
}

// ////////////////////////////////////////////////////////////////////
// Explicit template instantiation
namespace ba = boost::archive;
template void AirlineClassListKey::
serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
template void AirlineClassListKey::
serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
// ////////////////////////////////////////////////////////////////////

}
```

33.145 stdair/bom/AirlineClassListKey.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::AirlineClassListKey

  Key of airport-pair.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.146 stdair/bom/AirlineClassListKey.hpp

```
#ifndef __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
#define __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair  {

  struct AirlineClassListKey : public KeyAbstract {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  private:
    AirlineClassListKey();

  public:
    AirlineClassListKey (const AirlineCodeList_T&,
                         const ClassList_StringList_T&);

    AirlineClassListKey (const AirlineClassListKey&);

    ~AirlineClassListKey();


  public:
    // /////////// Getters //////////
    const AirlineCodeList_T& getAirlineCodeList() const {
      return _airlineCodeList;
    }

    const ClassList_StringList_T& getClassCodeList() const {
      return _classCodeList;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();

```

```
00110
00111   private:
00112     // ///////////////// Attributes ///////////////
00116     AirlineCodeList_T _airlineCodeList;
00117
00121     ClassList_StringList_T _classCodeList;
00122   };
00123
00124 }
00125 #endif // __STDAIR_BOM_AIRLINECLASSLISTKEY_HPP
```

33.147 stdair/bom/AirlineClassListTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< AirlineClassList * > stdair::AirlineClassListList_T
- typedef std::map< const MapKey_T, AirlineClassList * > stdair::AirlineClassListMap_T
- typedef std::pair< MapKey_T, AirlineClassList * > stdair::AirlineClassListWithKey_T
- typedef std::list< AirlineClassListWithKey_T > stdair::AirlineClassListDetailedList_T

33.148 stdair/bom/AirlineClassListTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00003 #define __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // STDAIR
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class AirlineClassList;
00018
00020   typedef std::list<AirlineClassList*> AirlineClassListList_T;
00021
00023   typedef std::map<const MapKey_T, AirlineClassList*> AirlineClassListMap_T;
00024
00026   typedef std::pair<MapKey_T, AirlineClassList*> AirlineClassListWithKey_T;
00027   typedef std::list<AirlineClassListWithKey_T> AirlineClassListDetailedList_T;
00028 }
00029 #endif // __STDAIR_BOM_AIRLINECLASSLISTTYPES_HPP
```

33.149 stdair/bom/AirlineFeature.cpp File Reference

```
#include <cassert>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.150 stdair/bom/AirlineFeature.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // StdAir
00007 #include <stdair/stdair_types.hpp>
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/AirlineFeature.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   AirlineFeature::AirlineFeature (const Key_T& iKey) :
00015     _key (iKey),
00016     _forecastingMethod(DEFAULT_FORECASTING_METHOD),
00017     _unconstrainingMethod(DEFAULT_UNCONSTRAINING_METHOD),
00018     _preOptimisationMethod(DEFAULT_PREOPTIMISATION_METHOD),
00019     _optimisationMethod(DEFAULT_OPTIMISATION_METHOD),
00020     _partnershipTechnique(DEFAULT_PARTNERSHIP_TECHNIQUE) {
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   AirlineFeature::AirlineFeature (const AirlineFeature& iAirlineFeature) :
00025     _key (iAirlineFeature._key),
00026     _forecastingMethod (iAirlineFeature._forecastingMethod),
00027     _unconstrainingMethod (iAirlineFeature._unconstrainingMethod),
00028     _preOptimisationMethod (iAirlineFeature._preOptimisationMethod),
00029     _optimisationMethod (iAirlineFeature._optimisationMethod),
00030     _partnershipTechnique (iAirlineFeature._partnershipTechnique) {
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   AirlineFeature::~AirlineFeature () {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   void AirlineFeature::init(const ForecastingMethod& iForecastingMethod,
00039                             const UnconstrainingMethod& iUnconstrainingMethod,
00040                             const PreOptimisationMethod& iPreOptimisationMethod,
00041                             const OptimisationMethod& iOptimisationMethod,
00042                             const HistoricalDataLimit_T& iHistoricalDataLimit,
00043                             const ControlMode_T& iControlMode,
00044                             const PartnershipTechnique& iPartnershipTechnique) {
00045     _forecastingMethod = iForecastingMethod;
00046     _unconstrainingMethod = iUnconstrainingMethod;
00047     _preOptimisationMethod = iPreOptimisationMethod;
00048     _optimisationMethod = iOptimisationMethod;
00049     _historicalDataLimit = iHistoricalDataLimit;
00050     _controlMode = iControlMode;
00051     _partnershipTechnique = iPartnershipTechnique;
00052   }
00053
00054   // ////////////////////////////////////////////////////////////////////
00055   std::string AirlineFeature::toString() const {
00056     std::ostringstream ostr;
00057     ostr << describeKey()
00058          << ", " << _forecastingMethod
00059          << ", " << _unconstrainingMethod
00060          << ", " << _preOptimisationMethod
00061          << ", " << _optimisationMethod
00062          << ", " << _historicalDataLimit
00063       //<< ", " << _controlMode
00064          << ", " << _partnershipTechnique;
00065     return ostr.str();
```

```
00066   }
00067
00068 }
00069
```

33.151 stdair/bom/AirlineFeature.hpp File Reference

```
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeatureKey.hpp>
#include <stdair/bom/AirlineFeatureTypes.hpp>
```

Classes

- class stdair::AirlineFeature

  Class representing various configuration parameters (e.g., revenue management methods such EMSRb or Monte-Carlo) for a given airline for the simulation.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.152 stdair/bom/AirlineFeature.hpp

```
00001 #ifndef __STDAIR_BOM_AIRLINEFEATURE_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATURE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_rm_types.hpp>
00009 #include <stdair/basic/UnconstrainingMethod.hpp>
00010 #include <stdair/basic/ForecastingMethod.hpp>
00011 #include <stdair/basic/PreOptimisationMethod.hpp>
00012 #include <stdair/basic/OptimisationMethod.hpp>
00013 #include <stdair/basic/PartnershipTechnique.hpp>
00014 #include <stdair/bom/BomAbstract.hpp>
00015 #include <stdair/bom/AirlineFeatureKey.hpp>
00016 #include <stdair/bom/AirlineFeatureTypes.hpp>
00017
00018 namespace stdair {
00019
00025   class AirlineFeature : public BomAbstract {
00026     template <typename BOM> friend class FacBom;
00027     template <typename BOM> friend class FacCloneBom;
00028     friend class FacBomManager;
00029
00030   public:
00031     // ////////////// Type definitions ///////////////
00035     typedef AirlineFeatureKey Key_T;
00036
00037   public:
00038     // /////////// Display support methods /////////
00044     void toStream (std::ostream& ioOut) const {
00045       ioOut << toString();
00046     }
00047
00053     void fromStream (std::istream& ioIn) {
00054     }
00055
00059     std::string toString() const;
00060
00064     const std::string describeKey() const {
00065       return _key.toString();
00066     }
00067
00068   public:
00069     // /////////// Getters ////////////
00073     const Key_T& getKey() const {
00074       return _key;
00075     }
00076
00080     BomAbstract* const getParent() const {
00081       return _parent;
00082     }
00083
00087     const  HolderMap_T& getHolderMap() const {
00088       return _holderMap;
00089     }
00090
00094     ForecastingMethod::EN_ForecastingMethod getForecastingMethod() const {
00095       return _forecastingMethod.getMethod();
00096     }
00097
00101     UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod() const
      {
00102       return _unconstrainingMethod.getMethod();
00103     }
```

```
00104
00108     PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique() const
      {
00109       return _partnershipTechnique.getTechnique();
00110     }
00111
00115     PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod() co
      nst {
00116       return _preOptimisationMethod.getMethod();
00117     }
00118
00122     OptimisationMethod::EN_OptimisationMethod getOptimisationMethod() const {
00123       return _optimisationMethod.getMethod();
00124     }
00125
00126
00127   public:
00128     // ///////////// Setters //////////
00139     void init (const ForecastingMethod&,
00140                const UnconstrainingMethod&,
00141                const PreOptimisationMethod&,
00142                const OptimisationMethod&,
00143                const HistoricalDataLimit_T&,
00144                const ControlMode_T&,
00145                const PartnershipTechnique&);
00146
00150     void setForecastingMethod (const ForecastingMethod& iForecastingMethod) {
00151       _forecastingMethod = iForecastingMethod;
00152     }
00153
00157     void setUnconstrainingMethod(const UnconstrainingMethod& iUnconstrainingMetho
      d) {
00158       _unconstrainingMethod = iUnconstrainingMethod;
00159     }
00160
00164     void setPartnershipTechnique(const PartnershipTechnique& iPartnershipTechniqu
      e) {
00165       _partnershipTechnique = iPartnershipTechnique;
00166     }
00167
00171     void setPreOptimisationMethod(const PreOptimisationMethod& iPreOptimisationMe
      thod) {
00172       _preOptimisationMethod = iPreOptimisationMethod;
00173     }
00174
00178     void setOptimisationMethod(const OptimisationMethod& iOptimisationMethod) {
00179       _optimisationMethod = iOptimisationMethod;
00180     }
00181
00182
00183   protected:
00184     // ////////// Constructors and destructors /////////
00188     AirlineFeature (const Key_T&);
00192     virtual ~AirlineFeature ();
00193
00194   private:
00198     AirlineFeature ();
00202      AirlineFeature (const AirlineFeature&);
00203
00204   protected:
00205     // ///////////// Attributes /////////////
00209     Key_T _key;
00210
00214     BomAbstract* _parent;
00215
00219     HolderMap_T _holderMap;
00220
```

```
    ForecastingMethod _forecastingMethod;

    HistoricalDataLimit_T _historicalDataLimit;

    ControlMode_T _controlMode;

    UnconstrainingMethod _unconstrainingMethod;

    PreOptimisationMethod _preOptimisationMethod;

    OptimisationMethod _optimisationMethod;

    PartnershipTechnique _partnershipTechnique;

  };

}
#endif // __STDAIR_BOM_AIRLINEFEATURE_HPP
```

33.153 stdair/bom/AirlineFeatureKey.cpp File Reference

```
#include <sstream>
#include <stdair/bom/AirlineFeatureKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.154 stdair/bom/AirlineFeatureKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
// StdAir
#include <stdair/bom/AirlineFeatureKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  AirlineFeatureKey::AirlineFeatureKey (const AirlineCode_T& iAirlineCode)
    : _airlineCode (iAirlineCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineFeatureKey::~AirlineFeatureKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineFeatureKey::toStream (std::ostream& ioOut) const {
    ioOut << "AirlineFeatureKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineFeatureKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string AirlineFeatureKey::toString() const {
    std::ostringstream oStr;
    oStr << _airlineCode;
    return oStr.str();
  }

}
```

33.155 stdair/bom/AirlineFeatureKey.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::AirlineFeatureKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.156 stdair/bom/AirlineFeatureKey.hpp

```
00001 #ifndef __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00002 #define __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/bom/KeyAbstract.hpp>
00012
00013 namespace stdair {
00015   struct AirlineFeatureKey : public KeyAbstract {
00016
00017   public:
00018     // /////////// Construction ///////////
00020     AirlineFeatureKey (const AirlineCode_T& iAirlineCode);
00021
00023     ~AirlineFeatureKey ();
00024
00025     // /////////// Getters //////////
00027     const AirlineCode_T& getAirlineCode() const { return _airlineCode; }
00028
00029     // /////////// Display support methods /////////
00032     void toStream (std::ostream& ioOut) const;
00033
00036     void fromStream (std::istream& ioIn);
00037
00043     const std::string toString() const;
00044
00045   private:
00046     // Attributes
00048     AirlineCode_T _airlineCode;
00049   };
00050
00051 }
00052 #endif // __STDAIR_BOM_AIRLINEFEATUREKEY_HPP
```

33.157 stdair/bom/AirlineFeatureTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< AirlineFeature * > stdair::AirlineFeatureList_T
- typedef std::map< const MapKey_T, AirlineFeature * > stdair::AirlineFeatureMap_T

33.158 stdair/bom/AirlineFeatureTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00003 #define __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class AirlineFeature;
00018
00020   typedef std::list<AirlineFeature*> AirlineFeatureList_T;
00021
00023   typedef std::map<const MapKey_T, AirlineFeature*> AirlineFeatureMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_AIRLINEFEATURETYPES_HPP
00027
```

33.159 stdair/bom/AirlineStruct.cpp File Reference

```
#include <cassert>
#include <istream>
#include <ostream>
#include <sstream>
#include <stdair/bom/AirlineStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.160 stdair/bom/AirlineStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <istream>
#include <ostream>
#include <sstream>
// StdAir
#include <stdair/bom/AirlineStruct.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  AirlineStruct::AirlineStruct () {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineStruct::AirlineStruct (const AirlineStruct& iAirlineStruct)
    : _code (iAirlineStruct._code), _name (iAirlineStruct._name) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineStruct::AirlineStruct (const AirlineCode_T& iAirlineCode,
                                const std::string& iAirlineName)
    : _code (iAirlineCode), _name (iAirlineName) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineStruct::~AirlineStruct () {
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void AirlineStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string AirlineStruct::describe() const {
    std::ostringstream oStr;
    oStr << _code << " " << _name;
    return oStr.str();
  }

}
```

33.161 stdair/bom/AirlineStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::AirlineStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.162 stdair/bom/AirlineStruct.hpp

```
00001 #ifndef __STDAIR_BOM_AIRLINESTRUCT_HPP
00002 #define __STDAIR_BOM_AIRLINESTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_inventory_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014
00015 namespace stdair {
00016
00018   struct AirlineStruct : public StructAbstract {
00019   public:
00020     // /////////// Getters ///////////////
00022     const AirlineCode_T& getAirlineCode() const {
00023       return _code;
00024     }
00025
00027     const std::string& getAirlineName() const {
00028       return _name;
00029     }
00030
00031     // /////////// Setters ///////////////
00033     void setAirlineCode (const AirlineCode_T& iAirlineCode) {
00034       _code = iAirlineCode;
00035     }
00036
00038     void setAirlineName (const std::string& iAirlineName) {
00039       _name = iAirlineName;
00040     }
00041
00042
00043   public:
00044     // /////////// Display support method /////////////
00047     void toStream (std::ostream& ioOut) const;
00048
00051     void fromStream (std::istream& ioIn);
00052
00054     const std::string describe() const;
00055
00056
00057   public:
00058     // ////////////// Constructors & Destructor ///////////////
00060     AirlineStruct (const AirlineCode_T&, const std::string& iAirlineName);
00062     AirlineStruct ();
00064     AirlineStruct (const AirlineStruct&);
00066     ~AirlineStruct ();
00067
00068
00069   private:
00070     // //////////////////// Attributes ////////////////////
00072     AirlineCode_T _code;
00073
00075     std::string _name;
00076   };
00077
00078 }
00079 #endif // __STDAIR_BOM_AIRLINESTRUCT_HPP
```

33.163 stdair/bom/AirportPair.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirportPair.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.164 stdair/bom/AirportPair.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/AirportPair.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  AirportPair::AirportPair()
    : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION),
      _parent (NULL)  {
    // That constructor is used by the serialisation process
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPair::AirportPair (const AirportPair& iAirportPair)
    : _key (iAirportPair.getKey()), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPair::AirportPair (const Key_T& iKey)
    : _key (iKey), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPair::~AirportPair () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string AirportPair::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }
}
```

33.165 stdair/bom/AirportPair.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirportPairKey.hpp>
#include <stdair/bom/AirportPairTypes.hpp>
```

Classes

- class stdair::AirportPair

  Class representing the actual attributes for an airport-pair.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.166 stdair/bom/AirportPair.hpp

```
00001 #ifndef __STDAIR_BOM_AIRPORTPAIR_HPP
00002 #define __STDAIR_BOM_AIRPORTPAIR_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/AirportPairKey.hpp>
00010 #include <stdair/bom/AirportPairTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018   class AirportPair : public BomAbstract {
00019     template <typename BOM> friend class FacBom;
00020     template <typename BOM> friend class FacCloneBom;
00021     friend class FacBomManager;
00022
00023   public:
00024     // ////////////// Type definitions //////////////
00028     typedef AirportPairKey Key_T;
00029
00030   public:
00031     // /////////// Display support methods /////////
00037     void toStream (std::ostream& ioOut) const {
00038       ioOut << toString();
00039     }
00040
00046     void fromStream (std::istream& ioIn) {
00047     }
00048
00052     std::string toString() const;
00053
00057     const std::string describeKey() const {
00058       return _key.toString();
00059     }
00060
00061   public:
00062     // /////////// Getters ////////////
00066     const Key_T& getKey() const {
00067       return _key;
00068     }
00069
00073     const AirportCode_T& getBoardingPoint() const {
00074       return _key.getBoardingPoint();
00075     }
00076
00080     const AirportCode_T& getOffPoint() const {
00081       return _key.getOffPoint();
00082     }
00083
00087     BomAbstract* const getParent() const {
00088       return _parent;
00089     }
00090
00094     const HolderMap_T& getHolderMap() const {
00095       return _holderMap;
00096     }
00097
00098   protected:
00099     // /////////// Constructors and destructors /////////
00103     AirportPair (const Key_T&);
00107     virtual ~AirportPair();
00108
```

```
  private:
    AirportPair();
    AirportPair (const AirportPair&);

  protected:
    // ############# Attributes #############
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;

  };

}
#endif // __STDAIR_BOM_AIRPORTPAIR_HPP
```

33.167 stdair/bom/AirportPairKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirportPairKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.168 stdair/bom/AirportPairKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// STDAIR
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/AirportPairKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  AirportPairKey::AirportPairKey ()
    : _boardingPoint (DEFAULT_ORIGIN),
      _offPoint (DEFAULT_DESTINATION) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPairKey::AirportPairKey (const AirportCode_T& iBoardingPoint,
                                  const AirportCode_T& iOffPoint)
    : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPairKey::AirportPairKey (const AirportPairKey& iKey)
    : _boardingPoint (iKey._boardingPoint),
      _offPoint (iKey._offPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPairKey::~AirportPairKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void AirportPairKey::toStream (std::ostream& ioOut) const {
    ioOut << "AirportPairKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void AirportPairKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string AirportPairKey::toString() const {
    std::ostringstream oStr;
    oStr << _boardingPoint << DEFAULT_KEY_SUB_FLD_DELIMITER
         << " " << _offPoint;
    return oStr.str();
  }

}
```

33.169 stdair/bom/AirportPairKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- struct stdair::AirportPairKey

  Key of airport-pair.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.170 stdair/bom/AirportPairKey.hpp

```
#ifndef __STDAIR_BOM_AIRPORTPAIRKEY_HPP
#define __STDAIR_BOM_AIRPORTPAIRKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_basic_types.hpp>

namespace stdair  {

  struct AirportPairKey : public KeyAbstract {

  public:
    // /////////// Construction ///////////
    AirportPairKey (const stdair::AirportCode_T&,
                    const stdair::AirportCode_T&);
    AirportPairKey (const AirportPairKey&);
    ~AirportPairKey ();
  private:
    AirportPairKey ();

  public:
    // /////////// Getters //////////
    const stdair::AirportCode_T& getBoardingPoint() const {
      return _boardingPoint;
    }

    const stdair::AirportCode_T& getOffPoint() const {
      return _offPoint;
    }

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

  private:
    // ///////////////// Attributes ///////////////////
    AirportCode_T _boardingPoint;

    AirportCode_T _offPoint;
  };

}
#endif // __SIMFQT_BOM_AIRPORTPAIRKEY_HPP
```

33.171 stdair/bom/AirportPairTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< AirportPair * > stdair::AirportPairList_T
- typedef std::map< const MapKey_T, AirportPair * > stdair::AirportPairMap_T
- typedef std::pair< MapKey_T, AirportPair * > stdair::AirportPairWithKey_T
- typedef std::list< AirportPairWithKey_T > stdair::AirportPairDetailedList_T

33.172 stdair/bom/AirportPairTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_AIRPORTPAIRTYPES_HPP
#define __STDAIR_BOM_AIRPORTPAIRTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class AirportPair;

  typedef std::list<AirportPair*> AirportPairList_T;

  typedef std::map<const MapKey_T, AirportPair*> AirportPairMap_T;

  typedef std::pair<MapKey_T, AirportPair*> AirportPairWithKey_T;
  typedef std::list<AirportPairWithKey_T> AirportPairDetailedList_T;
}
#endif // __STDAIR_BOM_AIRPORTPAIRTYPES_HPP

```

33.173 stdair/bom/BomAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <map>
#include <typeinfo>
```

Classes

- class stdair::BomAbstract

  Base class for the Business Object Model (BOM) layer.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::map< const std::type_info ∗, BomAbstract ∗ > stdair::HolderMap_T

Functions

- template<class charT , class traits >
  std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::BomAbstract &iBom)
- template<class charT , class traits >
  std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::BomAbstract &ioBom)

33.173.1 Function Documentation

33.173.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::BomAbstract & *iBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 74 of file BomAbstract.hpp.

33.173.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::BomAbstract & *ioBom*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by

Addison-Wesley.

Definition at line 102 of file BomAbstract.hpp.

References stdair::BomAbstract::fromStream().

33.174 stdair/bom/BomAbstract.hpp

```
00001
00007 #ifndef __STDAIR_BOM_BOMABSTRACT_HPP
00008 #define __STDAIR_BOM_BOMABSTRACT_HPP
00009
00010 // ////////////////////////////////////////////////////////////////////
00011 // Import section
00012 // ////////////////////////////////////////////////////////////////////
00013 // STL
00014 #include <iosfwd>
00015 #include <string>
00016 #include <map>
00017 #include <typeinfo>
00018
00019 namespace stdair {
00020
00024   class BomAbstract {
00025   public:
00026     // /////////// Display support methods /////////
00032     virtual void toStream (std::ostream& ioOut) const = 0;
00033
00039     virtual void fromStream (std::istream& ioIn) = 0;
00040
00046     virtual std::string toString() const = 0;
00047
00048
00049   protected:
00053     BomAbstract() {}
00054     BomAbstract(const BomAbstract&) {}
00055   public:
00059     virtual ~BomAbstract() {}
00060   };
00061
00062   /* Define the map of object holder type. */
00063   typedef std::map<const std::type_info*, BomAbstract*> HolderMap_T;
00064 }
00065
00071 template <class charT, class traits>
00072 inline
00073 std::basic_ostream<charT, traits>&
00074 operator<< (std::basic_ostream<charT, traits>& ioOut,
00075             const stdair::BomAbstract& iBom) {
00081   std::basic_ostringstream<charT,traits> ostr;
00082   ostr.copyfmt (ioOut);
00083   ostr.width (0);
00084
00085   // Fill string stream
00086   iBom.toStream (ostr);
00087
00088   // Print string stream
00089   ioOut << ostr.str();
00090
00091   return ioOut;
00092 }
00093
00099 template <class charT, class traits>
00100 inline
00101 std::basic_istream<charT, traits>&
00102 operator>> (std::basic_istream<charT, traits>& ioIn,
00103             stdair::BomAbstract& ioBom) {
00104   // Fill Bom object with input stream
00105   ioBom.fromStream (ioIn);
00106   return ioIn;
00107 }
00108
00109 #endif // __STDAIR_BOM_BOMABSTRACT_HPP
```

33.175 stdair/bom/BomArchive.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/tmpdir.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/base_object.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/list.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomArchive.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.176 stdair/bom/BomArchive.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/tmpdir.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/base_object.hpp>
#include <boost/serialization/utility.hpp>
#include <boost/serialization/list.hpp>
//#include <boost/serialization/assume_abstract.hpp>
// StdAir
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomArchive.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  void BomArchive::archive (const BomRoot& iBomRoot) {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BomArchive::archive (const Inventory& iInventory) {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << iInventory;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void BomArchive::restore (const std::string& iArchive,
                            Inventory& ioInventory) {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> ioInventory;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomArchive::archive (const FlightDate& iFlightDate) {
  }

}
```

33.177 stdair/bom/BomArchive.hpp File Reference

```
#include <iosfwd>
```

Classes

- class stdair::BomArchive

  Utility class to archive/restore BOM objects with Boost serialisation.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.178 stdair/bom/BomArchive.hpp

```
#ifndef __STDAIR_BOM_BOMARCHIVE_HPP
#define __STDAIR_BOM_BOMARCHIVE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>

namespace stdair {

  class BomRoot;
  class Inventory;
  class FlightDate;
  class LegDate;
  class SegmentDate;
  class LegCabin;
  class SegmentCabin;
  class FareFamily;
  class BookingClass;
  struct BookingRequestStruct;

  class BomArchive {
  public:
    static void archive (const BomRoot&);

    static std::string archive (const Inventory&);

    static void restore (const std::string& iArchive, Inventory&);

    static void archive (const FlightDate&);
  };

}
#endif // __STDAIR_BOM_BOMARCHIVE_HPP
```

33.179 stdair/bom/BomDisplay.cpp File Reference

33.180 stdair/bom/BomDisplay.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <ostream>
00011 // StdAir
00012 #include <stdair/basic/BasConst_BomDisplay.hpp>
00013 #include <stdair/bom/BomManager.hpp>
00014 #include <stdair/bom/BomRoot.hpp>
00015 #include <stdair/bom/Inventory.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/LegDate.hpp>
00018 #include <stdair/bom/SegmentDate.hpp>
00019 #include <stdair/bom/LegCabin.hpp>
00020 #include <stdair/bom/SegmentCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/BookingClass.hpp>
00023 #include <stdair/bom/AirportPair.hpp>
00024 #include <stdair/bom/PosChannel.hpp>
00025 #include <stdair/bom/DatePeriod.hpp>
00026 #include <stdair/bom/TimePeriod.hpp>
00027 #include <stdair/bom/FareFeatures.hpp>
00028 #include <stdair/bom/YieldFeatures.hpp>
00029 #include <stdair/bom/AirlineClassList.hpp>
00030 #include <stdair/bom/Bucket.hpp>
00031 #include <stdair/bom/TravelSolutionTypes.hpp>
00032 #include <stdair/bom/TravelSolutionStruct.hpp>
00033 #include <stdair/bom/BomDisplay.hpp>
00034 #include <stdair/bom/OnDDate.hpp>
00035
00036 namespace stdair {
00037
00043   struct FlagSaver {
00044   public:
00046     FlagSaver (std::ostream& oStream)
00047       : _oStream (oStream), _streamFlags (oStream.flags()) {
00048     }
00049
00051     ~FlagSaver() {
00052       // Reset formatting flags of the given output stream
00053       _oStream.flags (_streamFlags);
00054     }
00055
00056   private:
00058     std::ostream& _oStream;
00060     std::ios::fmtflags _streamFlags;
00061   };
00062
00063   // ////////////////////////////////////////////////////////////////////
00064   void BomDisplay::list (std::ostream& oStream, const BomRoot& iBomRoot,
00065                          const AirlineCode_T& iAirlineCode,
00066                          const FlightNumber_T& iFlightNumber) {
00067     // Save the formatting flags for the given STL output stream
00068     FlagSaver flagSaver (oStream);
00069
00070     // Check whether there are Inventory objects
00071     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00072       return;
00073     }
00074
00075     // Browse the inventories
00076     unsigned short invIdx = 1;
00077     const InventoryList_T& lInventoryList =
```

```
      BomManager::getList<Inventory> (iBomRoot);
    for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
         itInv != lInventoryList.end(); ++itInv, ++invIdx) {
      const Inventory* lInv_ptr = *itInv;
      assert (lInv_ptr != NULL);

      // Retrieve the inventory key (airline code)
      const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();

      // Display only the requested inventories
      if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
        // Get the list of flight-dates for that inventory
        list (oStream, *lInv_ptr, invIdx, iFlightNumber);
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::list (std::ostream& oStream, const Inventory& iInventory,
                         const unsigned short iInventoryIndex,
                         const FlightNumber_T& iFlightNumber) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Check whether there are FlightDate objects
    if (BomManager::hasMap<FlightDate> (iInventory) == false) {
      return;
    }

    //
    const AirlineCode_T& lAirlineCode = iInventory.getAirlineCode();
    oStream << iInventoryIndex << ". " << lAirlineCode << std::endl;

    // Browse the flight-dates
    unsigned short lCurrentFlightNumber = 0;
    unsigned short flightNumberIdx = 0;
    unsigned short departureDateIdx = 1;
    const FlightDateMap_T& lFlightDateList =
      BomManager::getMap<FlightDate> (iInventory);
    for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
         itFD != lFlightDateList.end(); ++itFD, ++departureDateIdx) {
      const FlightDate* lFD_ptr = itFD->second;
      assert (lFD_ptr != NULL);

      // Retrieve the key of the flight-date
      const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
      const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

      // Display only the requested flight number
      if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {
        //
        if (lCurrentFlightNumber != lFlightNumber) {
          lCurrentFlightNumber = lFlightNumber;
          ++flightNumberIdx; departureDateIdx = 1;
          oStream << "  " << iInventoryIndex << "." << flightNumberIdx << ". "
                  << lAirlineCode << lFlightNumber << std::endl;
        }

        oStream << "    " << iInventoryIndex << "." << flightNumberIdx
                << "." << departureDateIdx << ". "
                << lAirlineCode << lFlightNumber << " / " << lFlightDateDate
                << std::endl;
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
```

```
void BomDisplay::listAirportPairDateRange (std::ostream& oStream,
                                           const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  // Check whether there are AirportPair objects
  if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
    return;
  }

  const AirportPairList_T& lAirportPairList =
    BomManager::getList<AirportPair> (iBomRoot);
  for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
       itAir != lAirportPairList.end(); ++itAir ) {
    const AirportPair* lAir_ptr = *itAir;
    assert (lAir_ptr != NULL);

    // Check whether there are date-period objects
    assert (BomManager::hasList<DatePeriod> (*lAir_ptr) == true);

    // Browse the date-period objects
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (*lAir_ptr);

    for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
         itDP != lDatePeriodList.end(); ++itDP) {
      const DatePeriod* lDP_ptr = *itDP;
      assert (lDP_ptr != NULL);

      // Display the date-period object
      oStream << lAir_ptr->describeKey()
              <<" / " << lDP_ptr->describeKey() << std::endl;
    }

  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                             const BomRoot& iBomRoot) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << std::endl;
  oStream << "================================================================"
          << std::endl;
  oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
  oStream << "================================================================"
          << std::endl;

  // Check whether there are Inventory objects
  if (BomManager::hasList<Inventory> (iBomRoot) == false) {
    return;
  }

  // Browse the inventories
  const InventoryList_T& lInventoryList =
    BomManager::getList<Inventory> (iBomRoot);
  for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
       itInv != lInventoryList.end(); ++itInv) {
    const Inventory* lInv_ptr = *itInv;
    assert (lInv_ptr != NULL);

    // Display the inventory
    csvDisplay (oStream, *lInv_ptr);
  }
}
```

```
// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvDisplay (std::ostream& oStream,
                             const Inventory& iInventory) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
  oStream << "Inventory: " << iInventory.describeKey() << std::endl;
  oStream << "++++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;

  // Check whether there are FlightDate objects
  if (BomManager::hasList<FlightDate> (iInventory) == false) {
    return;
  }

  // Browse the flight-dates
  const FlightDateList_T& lFlightDateList =
    BomManager::getList<FlightDate> (iInventory);
  for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
       itFD != lFlightDateList.end(); ++itFD) {
    const FlightDate* lFD_ptr = *itFD;
    assert (lFD_ptr != NULL);

    // Display the flight-date
    csvDisplay (oStream, *lFD_ptr);
  }

  // Check if the inventory contains a list of partners

  if (BomManager::hasList<Inventory> (iInventory)){

    // Browse the partner's inventories
    const InventoryList_T& lPartnerInventoryList =
      BomManager::getList<Inventory> (iInventory);

    for (InventoryList_T::const_iterator itInv = lPartnerInventoryList.begin();
         itInv != lPartnerInventoryList.end(); ++itInv) {

      oStream << "-------------------------------------------------" << std::endl;
      oStream << "Partner inventory:" << std::endl;
      oStream << "-------------------------------------------------" << std::endl;
      const Inventory* lInv_ptr = *itInv;
      assert (lInv_ptr != NULL);

      // Display the inventory
      csvDisplay (oStream, *lInv_ptr);
    }
    oStream << "*****************************************" << std::endl;
    oStream << std::endl;
  }

  // Check if the inventory contains a list of O&D dates

  if (BomManager::hasList<OnDDate> (iInventory)){

    //Browse the O&Ds
    const OnDDateList_T& lOnDDateList =
      BomManager::getList<OnDDate> (iInventory);

    for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
         itOnD != lOnDDateList.end(); ++itOnD) {
      oStream << "*****************************************" << std::endl;
      oStream << "O&D-Date:" << std::endl;
```

```
        oStream << "----------" << std::endl;
        oStream << "Airline, Date, Origin-Destination, Segments, " << std::endl;

        const OnDDate* lOnDDate_ptr = *itOnD;
        assert (lOnDDate_ptr != NULL);

        // Display the O&D date
        csvDisplay (oStream, *lOnDDate_ptr);
      }
      oStream << "*****************************************" << std::endl;
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDisplay (std::ostream& oStream,
                               const OnDDate& iOnDDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    const AirlineCode_T& lAirlineCode = iOnDDate.getAirlineCode();
    const Date_T& lDate = iOnDDate.getDate();
    const AirportCode_T& lOrigin = iOnDDate.getOrigin();
    const AirportCode_T& lDestination = iOnDDate.getDestination();

    oStream << lAirlineCode <<", " << lDate << ", "<< lOrigin << "-"
            << lDestination << ", " << iOnDDate.describeKey() << ", "
            << std::endl;

    const StringDemandStructMap_T& lDemandInfoMap =
      iOnDDate.getDemandInfoMap();

    // Check if the map contains information.
    const bool isInfoMapEmpty = lDemandInfoMap.empty();
    if (isInfoMapEmpty) {
      return;
    }
    assert (lDemandInfoMap.empty() ==false);

    oStream << "----------" << std::endl;
    oStream << "Cabin-Class path, Demand mean, Demand std dev, Yield, "
            << std::endl;

    for (StringDemandStructMap_T::const_iterator itDI = lDemandInfoMap.begin();
         itDI != lDemandInfoMap.end(); ++itDI) {

      const std::string& lCabinClassPath = itDI->first;
      const YieldDemandPair_T lYieldDemandPair =
        itDI->second;
      const Yield_T lYield = lYieldDemandPair.first;
      const MeanStdDevPair_T lMeanStdDevPair =
        lYieldDemandPair.second;
      const MeanValue_T lDemandMean = lMeanStdDevPair.first;
      const StdDevValue_T lDemandStdDev = lMeanStdDevPair.second;

      oStream << lCabinClassPath << ", "
              << lDemandMean << ", "
              << lDemandStdDev << ", "
              << lYield << ", "
              << std::endl;
    }

  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDisplay (std::ostream& oStream,
                               const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
```

```
  FlagSaver flagSaver (oStream);

  const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
  oStream << "*****************************************" << std::endl;
  oStream << "FlightDate: " << lAirlineCode << iFlightDate.describeKey()
          << std::endl;
  oStream << "*****************************************" << std::endl;

  //
  csvSegmentDateDisplay (oStream, iFlightDate);
  //
  csvLegDateDisplay (oStream, iFlightDate);

  //
  csvLegCabinDisplay (oStream, iFlightDate);

  //
  csvBucketDisplay (oStream, iFlightDate);

  //
  csvFareFamilyDisplay (oStream, iFlightDate);

  //
  csvBookingClassDisplay (oStream, iFlightDate);
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvLegDateDisplay (std::ostream& oStream,
                                    const FlightDate& iFlightDate) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "*****************************************" << std::endl;
  oStream << "Leg-Dates:" << std::endl
          << "----------" << std::endl;
  oStream << "Flight, Leg, BoardDate, BoardTime, "
          << "OffDate, OffTime, Date Offset, Time Offset, Elapsed, "
          << "Distance, Capacity, " << std::endl;

  // Retrieve the key of the flight-date
  const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
  const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
  const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

  // Check whether there are LegDate objects
  if (BomManager::hasList<LegDate> (iFlightDate) == false) {
    return;
  }

  // Browse the leg-dates
  const LegDateList_T& lLegDateList =
    BomManager::getList<LegDate> (iFlightDate);
  for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
       itLD != lLegDateList.end(); ++itLD) {
    const LegDate* lLD_ptr = *itLD;
    assert (lLD_ptr != NULL);

    oStream << lAirlineCode << lFlightNumber << " "
            << lFlightDateDate << ", ";

    oStream << lLD_ptr->getBoardingPoint() << "-"
            << lLD_ptr->getOffPoint() << ", "
            << lLD_ptr->getBoardingDate() << ", "
            << lLD_ptr->getBoardingTime() << ", "
            << lLD_ptr->getOffDate() << ", "
            << lLD_ptr->getOffTime() << ", "
            << lLD_ptr->getElapsedTime() << ", "
```

```
              << lLD_ptr->getDateOffset().days() << ", "
              << lLD_ptr->getTimeOffset() << ", "
              << lLD_ptr->getDistance() << ", "
              << lLD_ptr->getCapacity() << ", " << std::endl;
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvSegmentDateDisplay (std::ostream& oStream,
                                          const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "SegmentDates:" << std::endl
            << "----------" << std::endl;
    oStream << "Flight, Segment, Date"
            << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

      // Retrieve the key of the segment-date, as well as its dates
      const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();
      oStream << lAirlineCode << lFlightNumber << " " << lFlightDateDate << ", "
              << lBoardPoint << "-" << lOffPoint << ", " << lSegmentDateDate << s
td::endl;

      // Check if the current segment has corresponding marketing segments.
      const bool hasMarketingSDList = BomManager::hasList<SegmentDate>(*lSD_ptr);

      if (hasMarketingSDList == true) {
        //
        const SegmentDateList_T& lMarketingSDList = BomManager::getList<SegmentDa
te>(*lSD_ptr);

        oStream << " *** Marketed by ";
        for (SegmentDateList_T::const_iterator itMarketingSD = lMarketingSDList.b
egin();
             itMarketingSD != lMarketingSDList.end(); ++itMarketingSD) {
          SegmentDate* lMarketingSD_ptr = *itMarketingSD;
          FlightDate* lMarketingFD_ptr = BomManager::getParentPtr<FlightDate>(*lM
arketingSD_ptr);
          Inventory* lMarketingInv_ptr = BomManager::getParentPtr<Inventory>(*lMa
rketingFD_ptr);
          oStream << lMarketingInv_ptr->toString() << lMarketingFD_ptr->toString(
) <<" * ";
        }
      }
```

```

      // Check if the current segment is operated by another segment date.
      const SegmentDate* lOperatingSD_ptr = lSD_ptr->getOperatingSegmentDate ();
      if (lOperatingSD_ptr != NULL) {

        const FlightDate* lOperatingFD_ptr = BomManager::getParentPtr<FlightDate>
    (*lOperatingSD_ptr);
        const Inventory* lOperatingInv_ptr = BomManager::getParentPtr<Inventory>(
    *lOperatingFD_ptr);
        oStream << " *** Operated by " << lOperatingInv_ptr->toString()
                << lOperatingFD_ptr->toString() << std::endl;
      }

      oStream << std::endl;
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvLegCabinDisplay (std::ostream& oStream,
                                       const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "LegCabins:" << std::endl
            << "----------" << std::endl;
    oStream << "Flight, Leg, Cabin, "
            << "OffedCAP, PhyCAP, RgdADJ, AU, UPR, SS, Staff, WL, Group, "
            << "CommSpace, AvPool, Avl, NAV, GAV, ACP, ETB, BidPrice, "
            << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
      return;
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
      BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
      const LegDate* lLD_ptr = *itLD;
      assert (lLD_ptr != NULL);

      // Retrieve the key of the leg-date, as well as its off point
      const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

      // Browse the leg-cabins
      const LegCabinList_T& lLegCabinList =
        BomManager::getList<LegCabin> (*lLD_ptr);
      for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
           itLC != lLegCabinList.end(); ++itLC) {
        const LegCabin* lLC_ptr = *itLC;
        assert (lLC_ptr != NULL);

        oStream << lAirlineCode << lFlightNumber << " "
                << lFlightDateDate << ", ";

        oStream << lBoardPoint << "-" << lOffPoint
                << " " << lLegDateDate << ", ";
```

```

        oStream << lLC_ptr->getCabinCode() << ", ";

        oStream << lLC_ptr->getOfferedCapacity() << ", "
                << lLC_ptr->getPhysicalCapacity() << ", "
                << lLC_ptr->getRegradeAdjustment() << ", "
                << lLC_ptr->getAuthorizationLevel() << ", "
                << lLC_ptr->getUPR() << ", "
                << lLC_ptr->getSoldSeat() << ", "
                << lLC_ptr->getStaffNbOfSeats() << ", "
                << lLC_ptr->getWLNbOfSeats() << ", "
                << lLC_ptr->getGroupNbOfSeats() << ", "
                << lLC_ptr->getCommittedSpace() << ", "
                << lLC_ptr->getAvailabilityPool() << ", "
                << lLC_ptr->getAvailability() << ", "
                << lLC_ptr->getNetAvailability() << ", "
                << lLC_ptr->getGrossAvailability() << ", "
                << lLC_ptr->getAvgCancellationPercentage() << ", "
                << lLC_ptr->getETB() << ", "
                << lLC_ptr->getCurrentBidPrice() << ", "
                << std::endl;
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvSegmentCabinDisplay (std::ostream& oStream,
                                           const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvFareFamilyDisplay (std::ostream& oStream,
                                         const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "SegmentCabins:" << std::endl
            << "--------------" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Bkgs, MIN, UPR, "
            << "CommSpace, AvPool, BP, " << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

      // Retrieve the key of the segment-date, as well as its dates
      const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
```

```
      const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

      // Browse the segment-cabins
      const SegmentCabinList_T& lSegmentCabinList =
        BomManager::getList<SegmentCabin> (*lSD_ptr);
      for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
           itSC != lSegmentCabinList.end(); ++itSC) {
        const SegmentCabin* lSC_ptr = *itSC;
        assert (lSC_ptr != NULL);

        // Retrieve the key of the segment-cabin
        const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

        // Check whether there are fare family objects
        if (BomManager::hasList<FareFamily> (*lSC_ptr) == false) {
          continue;
        }

        // Browse the fare families
        const FareFamilyList_T& lFareFamilyList =
          BomManager::getList<FareFamily> (*lSC_ptr);
        for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
             itFF != lFareFamilyList.end(); ++itFF) {
          const FareFamily* lFF_ptr = *itFF;
          assert (lFF_ptr != NULL);

          oStream << lAirlineCode << lFlightNumber << " "
                  << lFlightDateDate << ", ";

          oStream << lBoardPoint << "-" << lOffPoint << " "
                  << lSegmentDateDate << ", ";

          oStream << lCabinCode << ", " << lFF_ptr->getFamilyCode() << ", ";

          oStream << lSC_ptr->getBookingCounter() << ", "
                  << lSC_ptr->getMIN() << ", "
                  << lSC_ptr->getUPR() << ", "
                  << lSC_ptr->getCommittedSpace() << ", "
                  << lSC_ptr->getAvailabilityPool() << ", "
                  << lSC_ptr->getCurrentBidPrice() << ", "
                  << std::endl;
        }
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBucketDisplay (std::ostream& oStream,
                                     const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "*****************************************" << std::endl;
    oStream << "Buckets:" << std::endl
            << "--------" << std::endl;
    oStream << "Flight, Leg, Cabin, Yield, AU/SI, SS, AV, "
            << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
      return;
```

```
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
      BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
      const LegDate* lLD_ptr = *itLD;
      assert (lLD_ptr != NULL);

      // Retrieve the key of the leg-date, as well as its off point
      const Date_T& lLegDateDate = lLD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lLD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();

      // Browse the leg-cabins
      const LegCabinList_T& lLegCabinList =
        BomManager::getList<LegCabin> (*lLD_ptr);
      for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
           itLC != lLegCabinList.end(); ++itLC) {
        const LegCabin* lLC_ptr = *itLC;
        assert (lLC_ptr != NULL);

        // Check whether there are bucket objects
        if (BomManager::hasList<Bucket> (*lLC_ptr) == false) {
          continue;
        }

        // Retrieve the key of the leg-cabin
        const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();

        // Browse the buckets
        const BucketList_T& lBucketList = BomManager::getList<Bucket> (*lLC_ptr);

        for (BucketList_T::const_iterator itBuck = lBucketList.begin();
             itBuck != lBucketList.end(); ++itBuck) {
          const Bucket* lBucket_ptr = *itBuck;
          assert (lBucket_ptr != NULL);

          oStream << lAirlineCode << lFlightNumber << " "
                  << lFlightDateDate << ", ";

          oStream << lBoardPoint << "-" << lOffPoint << " "
                  << lLegDateDate << ", " << lCabinCode << ", ";

          oStream << lBucket_ptr->getYieldRangeUpperValue() << ", "
                  << lBucket_ptr->getSeatIndex() << ", "
                  << lBucket_ptr->getSoldSeats() << ", "
                  << lBucket_ptr->getAvailability() << ", ";
          oStream << std::endl;
        }
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
                                           const BookingClass& iBookingClass,
                                           const std::string& iLeadingString) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << iLeadingString << iBookingClass.getClassCode();

    if (iBookingClass.getSubclassCode() == 0) {
      oStream << ", ";
```

```
    } else {
      oStream << iBookingClass.getSubclassCode() << ", ";
    }
    oStream << iBookingClass.getAuthorizationLevel() << " ("
            << iBookingClass.getProtection() << "), "
            << iBookingClass.getNegotiatedSpace() << ", "
            << iBookingClass.getNoShowPercentage() << ", "
            << iBookingClass.getCancellationPercentage() << ", "
            << iBookingClass.getNbOfBookings() << ", "
            << iBookingClass.getNbOfGroupBookings() << " ("
            << iBookingClass.getNbOfPendingGroupBookings() << "), "
            << iBookingClass.getNbOfStaffBookings() << ", "
            << iBookingClass.getNbOfWLBookings() << ", "
            << iBookingClass.getETB() << ", "
            << iBookingClass.getNetClassAvailability() << ", "
            << iBookingClass.getNetRevenueAvailability() << ", "
            << iBookingClass.getSegmentAvailability() << ", "
            << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvBookingClassDisplay (std::ostream& oStream,
                                           const FlightDate& iFlightDate) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Headers
    oStream << "*****************************************" << std::endl;
    oStream << "Subclasses:" << std::endl
            << "-----------" << std::endl;
    oStream << "Flight, Segment, Cabin, FF, Subclass, MIN/AU (Prot), "
            << "Nego, NS%, OB%, "
            << "Bkgs, GrpBks (pdg), StfBkgs, WLBkgs, ETB, "
            << "ClassAvl, RevAvl, SegAvl, "
            << std::endl;

    // Retrieve the key of the flight-date
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

      // Retrieve the key of the segment-date, as well as its dates
      const Date_T& lSegmentDateDate = lSD_ptr->getBoardingDate();
      const AirportCode_T& lBoardPoint = lSD_ptr->getBoardingPoint();
      const AirportCode_T& lOffPoint = lSD_ptr->getOffPoint();

      // Browse the segment-cabins
      const SegmentCabinList_T& lSegmentCabinList =
        BomManager::getList<SegmentCabin> (*lSD_ptr);
      for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
           itSC != lSegmentCabinList.end(); ++itSC) {
        const SegmentCabin* lSC_ptr = *itSC;
        assert (lSC_ptr != NULL);

```

```
        // Retrieve the key of the segment-cabin
        const CabinCode_T& lCabinCode = lSC_ptr->getCabinCode();

        // Build the leading string to be displayed
        std::ostringstream oSCLeadingStr;
        oSCLeadingStr << lAirlineCode << lFlightNumber << " "
                      << lFlightDateDate << ", "
                      << lBoardPoint << "-" << lOffPoint << " "
                      << lSegmentDateDate << ", "
                      << lCabinCode << ", ";

        // Default Fare Family code, when there are no FF
        FamilyCode_T lFamilyCode ("NoFF");

        // Check whether there are FareFamily objects
        if (BomManager::hasList<FareFamily> (*lSC_ptr) == true) {

          // Browse the fare families
          const FareFamilyList_T& lFareFamilyList =
            BomManager::getList<FareFamily> (*lSC_ptr);
          for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
               itFF != lFareFamilyList.end(); ++itFF) {
            const FareFamily* lFF_ptr = *itFF;
            assert (lFF_ptr != NULL);

            // Retrieve the key of the segment-cabin
            lFamilyCode = lFF_ptr->getFamilyCode();

            // Complete the leading string to be displayed
            std::ostringstream oFFLeadingStr;
            oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

            // Browse the booking-classes
            const BookingClassList_T& lBookingClassList =
              BomManager::getList<BookingClass> (*lFF_ptr);
            for (BookingClassList_T::const_iterator itBC =
                   lBookingClassList.begin();
                 itBC != lBookingClassList.end(); ++itBC) {
              const BookingClass* lBC_ptr = *itBC;
              assert (lBC_ptr != NULL);

              //
              csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
            }
          }

          // Go on to the next segment-cabin
          continue;
        }
        assert (BomManager::hasList<FareFamily> (*lSC_ptr) == false);

        // The fare family code is a fake one ('NoFF'), and therefore
        // does not vary
        std::ostringstream oFFLeadingStr;
        oFFLeadingStr << oSCLeadingStr.str() << lFamilyCode << ", ";

        // Browse the booking-classes, directly from the segment-cabin object
        const BookingClassList_T& lBookingClassList =
          BomManager::getList<BookingClass> (*lSC_ptr);
        for (BookingClassList_T::const_iterator itBC =
               lBookingClassList.begin();
             itBC != lBookingClassList.end(); ++itBC) {
          const BookingClass* lBC_ptr = *itBC;
          assert (lBC_ptr != NULL);

          //
          csvBookingClassDisplay (oStream, *lBC_ptr, oFFLeadingStr.str());
```

```
        }
      }
    }
    oStream << "*****************************************" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::
  csvDisplay (std::ostream& oStream,
              const TravelSolutionList_T& iTravelSolutionList) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "Travel solutions:";
    unsigned short idx = 0;
    for (TravelSolutionList_T::const_iterator itTS =
           iTravelSolutionList.begin();
         itTS != iTravelSolutionList.end(); ++itTS, ++idx) {
      const TravelSolutionStruct& lTS = *itTS;

      oStream << std::endl;
      oStream << "    [" << idx << "] " << lTS.display();
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::
  csvDisplay (std::ostream& oStream,
              const DatePeriodList_T& iDatePeriodList) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    // Browse the date-period objects
    for (DatePeriodList_T::const_iterator itDP = iDatePeriodList.begin();
         itDP != iDatePeriodList.end(); ++itDP) {
      const DatePeriod* lDP_ptr = *itDP;
      assert (lDP_ptr != NULL);

      // Display the date-period object
      csvDateDisplay (oStream, *lDP_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvSimFQTAirRACDisplay (std::ostream& oStream,
                                           const BomRoot& iBomRoot) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << std::endl;
    oStream << "================================================================"
            << std::endl;
    oStream << "BomRoot: " << iBomRoot.describeKey() << std::endl;
    oStream << "================================================================"
            << std::endl;

    // Check whether there are airport-pair objects
    if (BomManager::hasList<AirportPair> (iBomRoot) == false) {
      return;
    }

    // Browse the airport-pair objects
    const AirportPairList_T& lAirportPairList =
      BomManager::getList<AirportPair> (iBomRoot);
    for (AirportPairList_T::const_iterator itAir = lAirportPairList.begin();
```

```
         itAir != lAirportPairList.end(); ++itAir ) {
      const AirportPair* lAir_ptr = *itAir;
      assert (lAir_ptr != NULL);

      // Display the airport pair object
      csvAirportPairDisplay (oStream, *lAir_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvAirportPairDisplay (std::ostream& oStream,
                                          const AirportPair& iAirportPair) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;
    oStream << "AirportPair: " << iAirportPair.describeKey() << std::endl;
    oStream << "++++++++++++++++++++++++++++++++++++++++++++++++" << std::endl;

    // Check whether there are date-period objects
    if (BomManager::hasList<DatePeriod> (iAirportPair) == false) {
      return;
    }

    // Browse the date-period objects
    const DatePeriodList_T& lDatePeriodList =
      BomManager::getList<DatePeriod> (iAirportPair);
    for (DatePeriodList_T::const_iterator itDP = lDatePeriodList.begin();
         itDP != lDatePeriodList.end(); ++itDP) {
      const DatePeriod* lDP_ptr = *itDP;
      assert (lDP_ptr != NULL);

      // Display the date-period object
      csvDateDisplay (oStream, *lDP_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::csvDateDisplay (std::ostream& oStream,
                                   const DatePeriod& iDatePeriod) {

    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "------------------------------------------" << std::endl;
    oStream << "DatePeriod: " << iDatePeriod.describeKey() << std::endl;
    oStream << "------------------------------------------" << std::endl;

    // Check whether there are pos-channel objects
    if (BomManager::hasList<PosChannel> (iDatePeriod) == false) {
      return;
    }

    // Browse the pos-channel objects
    const PosChannelList_T& lPosChannelList =
      BomManager::getList<PosChannel> (iDatePeriod);
    for (PosChannelList_T::const_iterator itPC = lPosChannelList.begin();
         itPC != lPosChannelList.end(); ++itPC) {
      const PosChannel* lPC_ptr = *itPC;
      assert (lPC_ptr != NULL);

      // Display the pos-channel object
      csvPosChannelDisplay (oStream, *lPC_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
```

```
void BomDisplay::csvPosChannelDisplay (std::ostream& oStream,
                                       const PosChannel& iPosChannel) {
  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "****************************************" << std::endl;
  oStream << "PosChannel: " << iPosChannel.describeKey() << std::endl;
  oStream << "****************************************" << std::endl;

  // Check whether there are time-period objects
  if (BomManager::hasList<TimePeriod> (iPosChannel) == false) {
    return;
  }

  // Browse the time-period objects
  const TimePeriodList_T& lTimePeriodList =
    BomManager::getList<TimePeriod> (iPosChannel);
  for (TimePeriodList_T::const_iterator itTP = lTimePeriodList.begin();
       itTP != lTimePeriodList.end(); ++itTP) {
    const TimePeriod* lTP_ptr = *itTP;
    assert (lTP_ptr != NULL);

    // Display the time-period object
    csvTimeDisplay (oStream, *lTP_ptr);
  }
}

// ////////////////////////////////////////////////////////////////////
void BomDisplay::csvTimeDisplay (std::ostream& oStream,
                                 const TimePeriod& iTimePeriod) {

  // Save the formatting flags for the given STL output stream
  FlagSaver flagSaver (oStream);

  oStream << "----------------------------------------" << std::endl;
  oStream << "TimePeriod: " << iTimePeriod.describeKey() << std::endl;
  oStream << "----------------------------------------" << std::endl;

  // Only one of the fare/yield feature list exists. Each of the following
  // two methods will check for the existence of the list. So, only the
  // existing list will be actually displayed.
  csvFeatureListDisplay<FareFeatures> (oStream, iTimePeriod);
  csvFeatureListDisplay<YieldFeatures> (oStream, iTimePeriod);
}

// ////////////////////////////////////////////////////////////////////
template <typename FEATURE_TYPE>
void BomDisplay::csvFeatureListDisplay (std::ostream& oStream,
                                        const TimePeriod& iTimePeriod) {

  // Check whether there are fare/yield-feature objects
  if (BomManager::hasList<FEATURE_TYPE> (iTimePeriod) == false) {
    return;
  }

  // Browse the fare/yield-feature objects
  typedef typename BomHolder<FEATURE_TYPE>::BomList_T FeaturesList_T;
  const FeaturesList_T& lFeaturesList =
    BomManager::getList<FEATURE_TYPE> (iTimePeriod);
  for (typename FeaturesList_T::const_iterator itFF = lFeaturesList.begin();
       itFF != lFeaturesList.end(); ++itFF) {
    const FEATURE_TYPE* lFF_ptr = *itFF;
    assert (lFF_ptr != NULL);

    // Display the fare-features object
    csvFeaturesDisplay (oStream, *lFF_ptr);
  }
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  template <typename FEATURE_TYPE>
  void BomDisplay::csvFeaturesDisplay (std::ostream& oStream,
                                       const FEATURE_TYPE& iFeatures) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "--------------------------------------" << std::endl;
    oStream << "Fare/yield-Features: " << iFeatures.describeKey() << std::endl;
    oStream << "--------------------------------------" << std::endl;

    // Check whether there are airlineClassList objects
    if (BomManager::hasList<AirlineClassList> (iFeatures) == false) {
      return;
    }

    // Browse the airlineClassList objects
    const AirlineClassListList_T& lAirlineClassListList =
      BomManager::getList<AirlineClassList> (iFeatures);
    for (AirlineClassListList_T::const_iterator itACL =
           lAirlineClassListList.begin();
         itACL != lAirlineClassListList.end(); ++itACL) {
      const AirlineClassList* lACL_ptr = *itACL;
      assert (lACL_ptr != NULL);

      // Display the airlineClassList object
      csvAirlineClassDisplay(oStream, *lACL_ptr);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomDisplay::
  csvAirlineClassDisplay  (std::ostream& oStream,
                           const AirlineClassList& iAirlineClassList) {
    // Save the formatting flags for the given STL output stream
    FlagSaver flagSaver (oStream);

    oStream << "-----------------------------------" << std::endl;
    oStream << "AirlineClassList: "
            << iAirlineClassList.describeKey() << std::endl;
    oStream << "-----------------------------------" << std::endl;
  }

}
```

33.181 stdair/bom/BomDisplay.hpp File Reference

```
#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Classes

- class stdair::BomDisplay

  Utility class to display StdAir objects with a pretty format.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.182 stdair/bom/BomDisplay.hpp

```
#ifndef __STDAIR_BOM_BOMDISPLAY_HPP
#define __STDAIR_BOM_BOMDISPLAY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// StdAir
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>

namespace stdair {

  class BomRoot;
  class Inventory;
  class FlightDate;
  class LegDate;
  class SegmentDate;
  class LegCabin;
  class SegmentCabin;
  class FareFamily;
  class BookingClass;
  class AirportPair;
  class PosChannel;
  class DatePeriod;
  class TimePeriod;
  class FareFeatures;
  class YieldFeatures;
  class AirlineClassList;
  class OnDDate;

  class BomDisplay {
  public:
    // /////////////// Display support methods /////////////////

    static void list (std::ostream&, const BomRoot&,
                      const AirlineCode_T& iAirlineCode = "all",
                      const FlightNumber_T& iFlightNumber = 0);

    static void list (std::ostream&, const Inventory&,
                      const unsigned short iInventoryIndex = 0,
                      const FlightNumber_T& iFlightNumber = 0);

    static void listAirportPairDateRange (std::ostream&,
                                          const BomRoot&);

    static void csvDisplay (std::ostream&, const BomRoot&);

    static void csvDisplay (std::ostream&, const Inventory&);

    static void csvDisplay (std::ostream&, const OnDDate&);

    static void csvDisplay (std::ostream&, const FlightDate&);

    static void csvLegDateDisplay (std::ostream&, const FlightDate&);

    static void csvSegmentDateDisplay (std::ostream&, const FlightDate&);

    static void csvLegCabinDisplay (std::ostream&, const FlightDate&);

    static void csvSegmentCabinDisplay (std::ostream&, const FlightDate&);

    static void csvFareFamilyDisplay (std::ostream&, const FlightDate&);

```

```
    static void csvBucketDisplay (std::ostream&, const FlightDate&);

    static void csvBookingClassDisplay (std::ostream&, const BookingClass&,
                                        const std::string& iLeadingString);
    static void csvBookingClassDisplay (std::ostream&, const FlightDate&);

    static void csvDisplay (std::ostream&, const TravelSolutionList_T&);

    static void csvDisplay (std::ostream&, const DatePeriodList_T&);

    static void csvSimFQTAirRACDisplay (std::ostream&, const BomRoot&);

    static void csvAirportPairDisplay (std::ostream&, const AirportPair&);

    static void csvDateDisplay (std::ostream&, const DatePeriod&);

    static void csvPosChannelDisplay (std::ostream&, const PosChannel&);

    static void csvTimeDisplay (std::ostream&, const TimePeriod&);

    template <typename FEATURE_TYPE>
    static void csvFeatureListDisplay (std::ostream& oStream, const TimePeriod&);

    template <typename FEATURE_TYPE>
    static void csvFeaturesDisplay (std::ostream& oStream, const FEATURE_TYPE&);

    static void csvAirlineClassDisplay (std::ostream&, const AirlineClassList&);
  };

}
#endif // __STDAIR_BOM_BOMDISPLAY_HPP
```

33.183 stdair/bom/BomHolder.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolderKey.hpp>
```

Classes

- class stdair::BomHolder< BOM >

  Class representing the holder of BOM object containers (list and map).

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.184 stdair/bom/BomHolder.hpp

```
#ifndef __STDAIR_BOM_BOMHOLDER_HPP
#define __STDAIR_BOM_BOMHOLDER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
#include <list>
#include <map>
// StdAir
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolderKey.hpp>

namespace stdair {

  template <typename BOM>
  class BomHolder : public stdair::BomAbstract {
    template <typename> friend class FacBom;
    friend class FacBomManager;

  public:
    // ///////////////// Type definitions ///////////////
    typedef stdair::BomHolderKey Key_T;

    typedef std::list<BOM*> BomList_T;

    typedef std::map<const MapKey_T, BOM*> BomMap_T;


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const {
      return "BomHolder";
    }

    const std::string describeKey() const {
      return "BomHolder";
    }

  protected:
    BomHolder();

    BomHolder (const BomHolder&);

    BomHolder (const Key_T& iKey) : _key (iKey) { }

    ~BomHolder() { };

  public:
    // /////////////// Attributes /////////////////
    Key_T _key;

    BomList_T _bomList;

    BomMap_T _bomMap;
```

```
  };

}
#endif // __STDAIR_BOM_BOMHOLDER_HPP
```

33.185 stdair/bom/BomHolderKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/bom/BomHolderKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.186 stdair/bom/BomHolderKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// StdAir
#include <stdair/bom/BomHolderKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BomHolderKey::BomHolderKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  BomHolderKey::~BomHolderKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void BomHolderKey::toStream (std::ostream& ioOut) const {
    ioOut << "BomHolderKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BomHolderKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BomHolderKey::toString() const {
    std::ostringstream oStr;
    oStr << " -- HOLDER -- ";
    return oStr.str();
  }

}
```

33.187 stdair/bom/BomHolderKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::BomHolderKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.188 stdair/bom/BomHolderKey.hpp

```
#ifndef __STDAIR_BOM_BOMHOLDERKEY_HPP
#define __STDAIR_BOM_BOMHOLDERKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/KeyAbstract.hpp>

namespace stdair {
  struct BomHolderKey : public KeyAbstract {

  public:
    // /////////// Construction ///////////
    BomHolderKey ();
    ~BomHolderKey ();

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

    const std::string describe() const;

  };

}
#endif // __STDAIR_BOM_BOMHOLDERKEY_HPP
```

33.189 stdair/bom/BomID.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- struct stdair::BomID< BOM >

  Class wrapper of bom ID (e.g. pointer to object).

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.190 stdair/bom/BomID.hpp

```
#ifndef __STDAIR_BOM_BOMID_HPP
#define __STDAIR_BOM_BOMID_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>

namespace stdair {

  template <typename BOM>
  struct BomID {

  public:
    // ////////// Getters ////////////
    BOM& getObject () const;

  public:
    // ////////// Constructors and destructors /////////
    BomID (BOM& iBOM);

    BomID (const BomID&);

    ~BomID();

  private:
    BomID();

  private:
    // /////////////// Attributes ////////////////
    BOM* _id;
  };

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> BomID<BOM>::BomID (BOM& iBOM): _id (&iBOM) { }

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> BomID<BOM>::BomID (const BomID& iBomID)
    : _id (iBomID._id) { }

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> BomID<BOM>::~BomID () { }

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> BOM& BomID<BOM>::getObject () const {
    assert (_id != NULL);
    return *_id;
  }
}
#endif // __STDAIR_BOM_BOMID_HPP
```

33.191 stdair/bom/BomIDTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/bom/BomID.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef struct BomID< BookingClass > stdair::BookingClassID_T
- typedef std::list< BookingClassID_T > stdair::BookingClassIDList_T

33.192 stdair/bom/BomIDTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOMIDTYPES_HPP
00003 #define __STDAIR_BOM_BOMIDTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013 #include <stdair/bom/BomID.hpp>
00014
00015 namespace stdair {
00016
00017   // Forward declarations.
00018   class BookingClass;
00019
00021   typedef struct BomID<BookingClass> BookingClassID_T;
00022
00024   typedef std::list<BookingClassID_T> BookingClassIDList_T;
00025 }
00026 #endif // __STDAIR_BOM_BOMIDTYPES_HPP
00027
```

33.193 stdair/bom/BomINIImport.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomINIImport.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef char bpt::ptree

33.194 stdair/bom/BomINIImport.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
#if BOOST_VERSION >= 104100
// Boost Property Tree
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/ini_parser.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/basic/BasFileMgr.hpp>
#include <stdair/bom/BomINIImport.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/service/Logger.hpp>

#if BOOST_VERSION >= 104100
namespace bpt = boost::property_tree;
#else // BOOST_VERSION >= 104100
namespace bpt {
  typedef char ptree;
}
#endif // BOOST_VERSION >= 104100

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  void BomINIImport::importINIConfig (ConfigHolderStruct& iConfigHolder,
                                      const ConfigINIFile& iConfigINIFile) {

    // Get the config file name.
    const stdair::Filename_T lFilename = iConfigINIFile.name();

    // Check that the file path given as input corresponds to an actual file
    const bool doesExistAndIsReadable =
      stdair::BasFileMgr::doesExistAndIsReadable (lFilename);
    if (doesExistAndIsReadable == false) {
      STDAIR_LOG_DEBUG ("The config input file '" << lFilename
                        << "' can not be retrieved on the file-system.");
      return;
    }
    STDAIR_LOG_DEBUG ("Load the config input file '" << lFilename
                      << "' content into the configuration holder.");

#if BOOST_VERSION >= 104100

    // Transform the INI file into a BOOST property tree.
    bpt::ptree pt;
    bpt::ini_parser::read_ini(lFilename, pt);
    // Add the property tree to the configuration structure.
    iConfigHolder.add(pt);

#endif // BOOST_VERSION >= 104100
  }

}
```

33.195 stdair/bom/BomINIImport.hpp File Reference

```
#include <string>
#include <stdair/stdair_file.hpp>
```

Classes

- class stdair::BomINIImport

  Utility class to import StdAir objects in a INI format.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.196 stdair/bom/BomINIImport.hpp

```
#ifndef __STDAIR_BOM_BOMINIIMPORT_HPP
#define __STDAIR_BOM_BOMINIIMPORT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_file.hpp>


namespace stdair {

  struct ConfigHolderStruct;

  class BomINIImport {
  public:
    // ///////////////// Import support methods /////////////////
    static void importINIConfig (ConfigHolderStruct&,
                                 const ConfigINIFile&);

  };
}
#endif // __STDAIR_BOM_BOMINIIMPORT_HPP
```

33.197 stdair/bom/BomJSONExport.cpp File Reference

```
#include <cassert>
#include <ostream>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/bom/BomJSONExport.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.198 stdair/bom/BomJSONExport.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <ostream>
00007 #if BOOST_VERSION >= 103400
00008 // Boost ForEach
00009 #include <boost/foreach.hpp>
00010 #endif // BOOST_VERSION >= 103400
00011 // StdAir
00012 #include <stdair/stdair_date_time_types.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/BomManager.hpp>
00015 #include <stdair/bom/BomRoot.hpp>
00016 #include <stdair/bom/Inventory.hpp>
00017 #include <stdair/bom/FlightDate.hpp>
00018 #include <stdair/bom/LegDate.hpp>
00019 #include <stdair/bom/SegmentDate.hpp>
00020 #include <stdair/bom/LegCabin.hpp>
00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/FareFamily.hpp>
00023 #include <stdair/bom/BookingClass.hpp>
00024 #include <stdair/bom/Bucket.hpp>
00025 #include <stdair/bom/EventStruct.hpp>
00026 #include <stdair/bom/EventTypes.hpp>
00027 #include <stdair/bom/BookingRequestStruct.hpp>
00028 #include <stdair/bom/BreakPointStruct.hpp>
00029 #include <stdair/bom/BomJSONExport.hpp>
00030
00031 namespace stdair {
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   void BomJSONExport::
00035   jsonExportFlightDateList (std::ostream& oStream,
00036                             const BomRoot& iBomRoot,
00037                             const AirlineCode_T& iAirlineCode,
00038                             const FlightNumber_T& iFlightNumber) {
00039
00040     // Check whether there are Inventory objects
00041     if (BomManager::hasList<Inventory> (iBomRoot) == false) {
00042       return;
00043     }
00044
00045 #if BOOST_VERSION >= 104100
00046
00047     // Create empty property tree objects
00048     bpt::ptree pt;
00049     bpt::ptree ptInventoryList;
00050
00051     // Browse the inventories
00052     const InventoryList_T& lInventoryList =
00053       BomManager::getList<Inventory> (iBomRoot);
00054     for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00055          itInv != lInventoryList.end(); ++itInv) {
00056       const Inventory* lInv_ptr = *itInv;
00057       assert (lInv_ptr != NULL);
00058
00059       // Retrieve the inventory key (airline code)
00060       const AirlineCode_T& lAirlineCode = lInv_ptr->getAirlineCode();
00061
00062       // Display only the requested inventories
00063       if (iAirlineCode == "all" || iAirlineCode == lAirlineCode) {
00064
00065         // Flight date tree
```

```
        bpt::ptree ptFD;
        // Create an empty flight-dates array
        bpt::ptree lFDDatePropertyTree;

        // Check whether there are FlightDate objects
        if (BomManager::hasMap<FlightDate> (*lInv_ptr) == false) {
          return;
        }

        // Browse the flight-dates
        const FlightDateMap_T& lFlightDateList =
          BomManager::getMap<FlightDate> (*lInv_ptr);
        for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
             itFD != lFlightDateList.end(); ++itFD) {
          const FlightDate* lFD_ptr = itFD->second;
          assert (lFD_ptr != NULL);

          // Retrieve the key of the flight-date
          const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
          const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

          // Display only the requested flight number
          if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {

            // Add the airline code to the inventory tree
            ptFD.put ("airline_code", lAirlineCode);
            // Put flight number in property tree
            ptFD.put ("number", lFlightNumber);
            // Put flight date date in property tree
            ptFD.put ("date", lFlightDateDate);

            // Put the current flight date tree in the array
            ptInventoryList.push_back(std::make_pair("", ptFD));

          }
        }

      }
    }

    // Store the inventory(ies) array tree into the global tree
    pt.add_child ("inventories", ptInventoryList);

    // Write the property tree into the JSON stream.
    write_json (oStream, pt);
#endif // BOOST_VERSION >= 104100
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportFlightDate (bpt::ptree& ioFDPropertyTree,
                                            const Inventory& iInventory,
                                            const FlightNumber_T& iFlightNumber)
    {

    // Check whether there are FlightDate objects
    if (BomManager::hasMap<FlightDate> (iInventory) == false) {
      return;
    }

#if BOOST_VERSION >= 104100

    // Create an empty flight-dates array
    bpt::ptree lFDDatePropertyTree;

    // Browse the flight-dates
    const FlightDateMap_T& lFlightDateList =
      BomManager::getMap<FlightDate> (iInventory);
```

```
    for (FlightDateMap_T::const_iterator itFD = lFlightDateList.begin();
         itFD != lFlightDateList.end(); ++itFD) {
      const FlightDate* lFD_ptr = itFD->second;
      assert (lFD_ptr != NULL);

      // Retrieve the key of the flight-date
      const FlightNumber_T& lFlightNumber = lFD_ptr->getFlightNumber();
      const Date_T& lFlightDateDate = lFD_ptr->getDepartureDate();

      // Display only the requested flight number
      if (iFlightNumber == 0 || iFlightNumber == lFlightNumber) {

        // Create an empty property tree object for the current flight date
        bpt::ptree lCurrFDTree;

        // Put flight number in property tree
        lCurrFDTree.put ("number", lFlightNumber);
        // Put flight date date in property tree
        lCurrFDTree.put ("date", lFlightDateDate);

        // Put the current flight date tree in the flight date array
        ioFDPropertyTree.push_back(std::make_pair("", lCurrFDTree));

      }
    }
#endif // BOOST_VERSION >= 104100

  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::
  jsonExportFlightDateObjects (std::ostream& oStream,
                               const FlightDate& iFlightDate) {

#if BOOST_VERSION >= 104100

    // Create an empty property tree object
    bpt::ptree pt;

    // Put the airline code in property tree
    const AirlineCode_T& lAirlineCode = iFlightDate.getAirlineCode();
    pt.put ("flight_date.airline_code", lAirlineCode);

    // Put the flight number in property tree
    const FlightNumber_T& lFlightNumber = iFlightDate.getFlightNumber();
    pt.put ("flight_date.flight_number", lFlightNumber);

    // Put the flight departure date in property tree
    const Date_T& lFlightDateDate = iFlightDate.getDepartureDate();
    const std::string& lDepartureDateStr =
      boost::gregorian::to_simple_string (lFlightDateDate);
    pt.put ("flight_date.departure_date", lDepartureDateStr);

    // Create an empty legs array
    bpt::ptree ptLegs;

    // Recursively construct the legs array
    jsonExportLegDate (ptLegs, iFlightDate);

    // Add legs tree to the global property tree
    pt.add_child ("flight_date.legs", ptLegs);

    // Create an empty segments array
    bpt::ptree ptSegments;

    // Recursively construct the segments array
    jsonExportSegmentDate (ptSegments, iFlightDate);
```

```

    // Add segments tree to the global property tree
    pt.add_child ("flight_date.segments", ptSegments);

    // Write the property tree into the JSON stream.
    write_json (oStream, pt);

#endif // BOOST_VERSION >= 104100
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportLegDate (bpt::ptree& ioLegDateListTree,
                                         const FlightDate& iFlightDate) {

    // Check whether there are LegDate objects
    if (BomManager::hasList<LegDate> (iFlightDate) == false) {
      return;
    }

    // Browse the leg-dates
    const LegDateList_T& lLegDateList =
      BomManager::getList<LegDate> (iFlightDate);
    for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
         itLD != lLegDateList.end(); ++itLD) {
      const LegDate* lLD_ptr = *itLD;
      assert (lLD_ptr != NULL);

#if BOOST_VERSION >= 104100

      // Create an empty property tree object for the current leg date
      bpt::ptree lCurrLDTree;

      // Put boarding point in property tree
      const AirportCode_T& lBoardingPoint = lLD_ptr->getBoardingPoint();
      lCurrLDTree.put ("board_point", lBoardingPoint);
      // Put off point in property tree
      const AirportCode_T& lOffPoint = lLD_ptr->getOffPoint();
      lCurrLDTree.put ("off_point", lOffPoint);
      // Put boarding date in property tree
      const Date_T& lBoardingDate = lLD_ptr->getBoardingDate();
      lCurrLDTree.put ("board_date", lBoardingDate);
      // Put off date in property tree
      const Date_T& lOffDate = lLD_ptr->getOffDate();
      lCurrLDTree.put ("off_dDate", lOffDate);
      // Put boarding time in property tree
      const Duration_T& lBoardingTime = lLD_ptr->getBoardingTime();
      lCurrLDTree.put ("board_time", lBoardingTime);
      // Put off time in property tree
      const Duration_T& lOffTime = lLD_ptr->getOffTime();
      lCurrLDTree.put ("off_time", lOffTime);
      // Put elapsed time in property tree
      const Duration_T& lElapsedTime = lLD_ptr->getElapsedTime();
      lCurrLDTree.put ("elapsed_time", lElapsedTime);
      // Put date offset in property tree
      const DateOffset_T& lDateOffset = lLD_ptr->getDateOffset();
      lCurrLDTree.put ("date_offset", lDateOffset);
      // Put time offset in property tree
      const Duration_T& lTimeOffset = lLD_ptr->getTimeOffset();
      lCurrLDTree.put ("time_offset", lTimeOffset);
      // Put distance in property tree
      const Distance_T& lDistance = lLD_ptr->getDistance();
      lCurrLDTree.put ("distance", lDistance);
      // Put capacity in property tree
      const CabinCapacity_T& lCapacity = lLD_ptr->getCapacity();
      lCurrLDTree.put ("capacity", lCapacity);

      // Create an empty property tree object for the leg cabins array
```

```
      // corresponding to the current leg date.
      bpt::ptree lLegCabinArray;

      // Recursively construct the leg cabins array
      jsonExportLegCabin (lLegCabinArray, *lLD_ptr);

      // Add the leg cabins array to the leg date tree
      lCurrLDTree.add_child ("cabins", lLegCabinArray);

      // Put the current leg date tree in the leg date list tree
      ioLegDateListTree.push_back(std::make_pair("", lCurrLDTree));

#endif // BOOST_VERSION >= 104100
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportLegCabin (bpt::ptree& ioLegCabinListTree,
                                          const LegDate& iLegDate) {

    // Check whether there are LegCabin objects
    if (BomManager::hasList<LegCabin> (iLegDate) == false) {
      return;
    }

    // Browse the leg-cabins
    const LegCabinList_T& lLegCabinList =
      BomManager::getList<LegCabin> (iLegDate);
    for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
         itLC != lLegCabinList.end(); ++itLC) {
      const LegCabin* lLC_ptr = *itLC;
      assert (lLC_ptr != NULL);

#if BOOST_VERSION >= 104100

      // Create an empty property tree object for the current leg cabin
      bpt::ptree lCurrLCTree;
      bpt::ptree lCurrLCBPV;

      // Put the cabin code in property tree
      const CabinCode_T& lCabinCode = lLC_ptr->getCabinCode();
      lCurrLCTree.put ("code", lCabinCode);
      // Put the offered capacity in property tree
      const CabinCapacity_T& lOfferedCapacity = lLC_ptr->getOfferedCapacity();
      lCurrLCTree.put ("offed_cap", lOfferedCapacity);
      // Put the physical capacity in property tree
      const CabinCapacity_T& lPhysicalCapacity = lLC_ptr->getPhysicalCapacity();
      lCurrLCTree.put ("phy_cap", lPhysicalCapacity);
      // Put regrade adjustment in property tree
      const CapacityAdjustment_T& lRegradeAdjustment = lLC_ptr->getRegradeAdjustment();
      lCurrLCTree.put ("rgd_adj", lRegradeAdjustment);
      // Put authorization level in property tree
      const AuthorizationLevel_T& lAuthorizationLevel = lLC_ptr->getAuthorizationLevel();
      lCurrLCTree.put ("au", lAuthorizationLevel);
      // Put UPR in property tree
      const UPR_T& lUPR = lLC_ptr->getUPR();
      lCurrLCTree.put ("upr", lUPR);
      // Put sold seats in property tree
      const NbOfSeats_T& lNbOfSoldSeats = lLC_ptr->getSoldSeat();
      lCurrLCTree.put ("ss", lNbOfSoldSeats);
      // Put staff nb of seats in property tree
      const NbOfSeats_T& lStaffNbOfSeats = lLC_ptr->getStaffNbOfSeats();
      lCurrLCTree.put ("staff", lStaffNbOfSeats);
      // Put waiting list nb of seats in property tree
      const NbOfSeats_T& lWLNbOfSeats = lLC_ptr->getWLNbOfSeats();
```

```
      lCurrLCTree.put ("wl", lWLNbOfSeats);
      // Put group nb of seats in property tree
      const NbOfSeats_T& lGroupNbOfSeats = lLC_ptr->getGroupNbOfSeats();
      lCurrLCTree.put ("group", lGroupNbOfSeats);
      // Put committed space in property tree
      const CommittedSpace_T& lCommittedSpace = lLC_ptr->getCommittedSpace();
      lCurrLCTree.put ("comm_space", lCommittedSpace);
      // Put availability pool in property tree
      const Availability_T& lAvailabilityPool = lLC_ptr->getAvailabilityPool();
      lCurrLCTree.put ("av_pool", lAvailabilityPool);
      // Put availability in property tree
      const Availability_T& lAvailability = lLC_ptr->getAvailability();
      lCurrLCTree.put ("avl", lAvailability);
      // Put net availability in property tree
      const Availability_T& lNetAvailability = lLC_ptr->getNetAvailability();
      lCurrLCTree.put ("nav", lNetAvailability);
      // Put gross availability in property tree
      const Availability_T& lGrossAvailability = lLC_ptr->getGrossAvailability();

      lCurrLCTree.put ("gav", lGrossAvailability);
      // Put avg cancellation percentage in property tree
      const OverbookingRate_T& lAvgCancellationPercentage =
        lLC_ptr->getAvgCancellationPercentage();
      lCurrLCTree.put ("acp", lAvgCancellationPercentage);
      // Put ETB in property tree
      const NbOfSeats_T& lExpectedToBoard = lLC_ptr->getETB();
      lCurrLCTree.put ("etb", lExpectedToBoard );
      // Put current bid price in property tree
      const BidPrice_T& lCurrentBidPrice = lLC_ptr->getCurrentBidPrice();
      lCurrLCTree.put ("bid_price", lCurrentBidPrice);
      // Put current bid price vector in property tree
      const BidPriceVector_T& lCurrentBidPriceVector =
        lLC_ptr->getBidPriceVector();
      std::ostringstream ostr;
      BidPriceVector_T::const_iterator itBP = lCurrentBidPriceVector.begin();
      while (itBP != lCurrentBidPriceVector.end()) {
        ostr << *itBP;
        ++itBP;
        if (itBP != lCurrentBidPriceVector.end()) {
          ostr << ",";
        }
      }
      lCurrLCTree.put ("BPV", ostr.str());

      // Create an empty property tree object for the buckets array
      // corresponding to the current leg cabin.
      bpt::ptree lBucketTree;

      // Recursively construct the buckets array
      jsonExportBucket (lBucketTree, *lLC_ptr);

      // Add the buckets array to the leg cabin tree
      lCurrLCTree.add_child ("buckets", lBucketTree);

      // Put the current leg cabin tree in the leg cabin list tree
      ioLegCabinListTree.push_back(std::make_pair("", lCurrLCTree));

#endif // BOOST_VERSION >= 104100
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportBucket (bpt::ptree& ioBucketListTree,
                                        const LegCabin& iLegCabin) {

    // Check whether there are Bucket objects
    if (BomManager::hasList<Bucket> (iLegCabin) == false) {
```

```
      return;
    }

    // Browse the buckets
    const BucketList_T& lBucketList = BomManager::getList<Bucket> (iLegCabin);
    for (BucketList_T::const_iterator itBuck = lBucketList.begin();
         itBuck != lBucketList.end(); ++itBuck) {
      const Bucket* lBucket_ptr = *itBuck;
      assert (lBucket_ptr != NULL);

#if BOOST_VERSION >= 104100

      // Create an empty property tree object for the current bucket
      bpt::ptree lCurrBucketTree;

      // Put yield in property tree
      const Yield_T& lYieldRangeUpperValue =
        lBucket_ptr->getYieldRangeUpperValue();
      lCurrBucketTree.put ("yield", lYieldRangeUpperValue);
      // Put seat_index in property tree
      const SeatIndex_T& lSeatIndex = lBucket_ptr->getSeatIndex();
      lCurrBucketTree.put ("si", lSeatIndex);
      // Put sold_seats in property tree
      const NbOfSeats_T& lSoldSeats = lBucket_ptr->getSoldSeats();
      lCurrBucketTree.put ("ss", lSoldSeats);
      // Put avaibility in property tree
      const CabinCapacity_T& lAvailability = lBucket_ptr->getAvailability();
      lCurrBucketTree.put ("av", lAvailability);

      // Put the current bucket tree in the bucket list tree
      ioBucketListTree.push_back(std::make_pair("", lCurrBucketTree));

#endif // BOOST_VERSION >= 104100
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportSegmentDate (bpt::ptree& ioSegmentDateTree,
                                             const FlightDate& iFlightDate) {

    // Check whether there are SegmentDate objects
    if (BomManager::hasList<SegmentDate> (iFlightDate) == false) {
      return;
    }

    // Browse the segment-dates
    const SegmentDateList_T& lSegmentDateList =
      BomManager::getList<SegmentDate> (iFlightDate);
    for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
         itSD != lSegmentDateList.end(); ++itSD) {
      const SegmentDate* lSD_ptr = *itSD;
      assert (lSD_ptr != NULL);

#if BOOST_VERSION >= 104100

      // Create an empty property tree object for the current segment date
      bpt::ptree lCurrSDTree;

      // Put segment key in property tree
      lCurrSDTree.put ("segment", lSD_ptr->toString());

      // Create an empty property tree object for the segment cabin array
      // corresponding to the current segment date
      bpt::ptree lSegmentCabinTree;

      // Recursively construct the segment cabin array
      jsonExportSegmentCabin (lSegmentCabinTree, *lSD_ptr);
```

```
00477
00478       // Add the segment cabin array to the tree of the current segment date
00479       lCurrSDTree.add_child ("sub_classes", lSegmentCabinTree);
00480
00481       // Put segment date array in property tree
00482       ioSegmentDateTree.push_back(std::make_pair("", lCurrSDTree));
00483
00484 #endif // BOOST_VERSION >= 104100
00485     }
00486   }
00487
00488   // ////////////////////////////////////////////////////////////////////
00489   void BomJSONExport::jsonExportSegmentCabin (bpt::ptree& ioPropertyTree,
00490                                               const SegmentDate& iSegmentDate) {
00491
00492     // Check whether there are SegmentCabin objects
00493     if (BomManager::hasList<SegmentCabin> (iSegmentDate) == false) {
00494       return;
00495     }
00496
00497     // Browse the segment-cabins
00498     const SegmentCabinList_T& lSegmentCabinList =
00499       BomManager::getList<SegmentCabin> (iSegmentDate);
00500     for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
00501          itSC != lSegmentCabinList.end(); ++itSC) {
00502       const SegmentCabin* lSC_ptr = *itSC;
00503       assert (lSC_ptr != NULL);
00504
00505 #if BOOST_VERSION >= 104100
00506       // Create an empty property tree object for the current segment cabin
00507       bpt::ptree lSCArray;
00508
00509       // Put cabin in property tree
00510
00511       lSCArray.put ("cabin_code",lSC_ptr->toString());
00512
00513       // Export the cabin tree to add fare-families and sub-classes details
00514       jsonExportFareFamily (ioPropertyTree, lSCArray, *lSC_ptr);
00515
00516 #endif // BOOST_VERSION >= 104100
00517
00518     }
00519   }
00520
00521   // ////////////////////////////////////////////////////////////////////
00522   void BomJSONExport::jsonExportFareFamily (bpt::ptree& ioPropertyTree,
00523                                             bpt::ptree& ioSCTree,
00524                                             const SegmentCabin& iSegmentCabin) {

00525
00526     // Check whether there are FareFamily objects
00527     if (BomManager::hasList<FareFamily> (iSegmentCabin) == true) {
00528
00529       // Browse the fare-families
00530       const FareFamilyList_T& lFareFamilyList =
00531         BomManager::getList<FareFamily> (iSegmentCabin);
00532       for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
00533          itFF != lFareFamilyList.end(); ++itFF) {
00534         const FareFamily* lFF_ptr = *itFF;
00535         assert (lFF_ptr != NULL);
00536
00537         // Browse the booking-classes
00538         const BookingClassList_T& lBookingClassList =
00539           BomManager::getList<BookingClass> (*lFF_ptr);
00540         for (BookingClassList_T::const_iterator itBC =
00541                lBookingClassList.begin();
00542              itBC != lBookingClassList.end(); ++itBC) {
```

```
          const BookingClass* lBC_ptr = *itBC;
          assert (lBC_ptr != NULL);

#if BOOST_VERSION >= 104100

          // Put family code in property tree
          const FamilyCode_T& lFamilyCode = lFF_ptr->getFamilyCode();
          ioSCTree.put ("family_code", lFamilyCode);

          // Export the cabin tree to add sub-classes details
          jsonExportBookingClass (ioPropertyTree, ioSCTree, *lBC_ptr);

#endif // BOOST_VERSION >= 104100

        }
      }
    } else {

      // The fare family code is a fake one ('NoFF'), and therefore
      // does not vary
      const FamilyCode_T lDefaultFamilyCode ("NoFF");

      // Browse the booking-classes, directly from the segment-cabin object
      const BookingClassList_T& lBookingClassList =
        BomManager::getList<BookingClass> (iSegmentCabin);
      for (BookingClassList_T::const_iterator itBC =
             lBookingClassList.begin();
           itBC != lBookingClassList.end(); ++itBC) {
        const BookingClass* lBC_ptr = *itBC;
        assert (lBC_ptr != NULL);

#if BOOST_VERSION >= 104100

        // Put family code in property tree
        ioSCTree.put ("family_code", lDefaultFamilyCode);

        // Export the cabin tree to add sub-classes details
        jsonExportBookingClass (ioPropertyTree, ioSCTree, *lBC_ptr);

#endif // BOOST_VERSION >= 104100
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::jsonExportBookingClass (bpt::ptree& ioPropertyTree,
                                              bpt::ptree& ioSCTree,
                                              const BookingClass& iBookingClass)
  {

#if BOOST_VERSION >= 104100

    // Put sub class in property tree
    ioSCTree.put ("class_code", iBookingClass.toString());
    // Put authorization level in property tree
    std::ostringstream oAUBlStr;
    oAUBlStr << iBookingClass.getAuthorizationLevel();
      //<< " (" << iBookingClass.getCumulatedBookingLimit()
      //<< ") ";
    ioSCTree.put ("au", oAUBlStr.str());
    // Put negotiated space in property tree
    const NbOfSeats_T& lNegotiatedSpace =
      iBookingClass.getNegotiatedSpace();
    ioSCTree.put ("nego", lNegotiatedSpace);
    // Put no show percentage in property tree
    const OverbookingRate_T& lNoShowPercentage =
      iBookingClass.getNoShowPercentage();
```

```
    ioSCTree.put ("ns%", lNoShowPercentage);
    // Put cancellation percentage in property tree
    const OverbookingRate_T& lCancellationPercentage =
      iBookingClass.getCancellationPercentage();
    ioSCTree.put ("ob%", lCancellationPercentage);
    // Put sub nb of bookings in property tree
    const NbOfBookings_T lNbOfBookings =
      iBookingClass.getNbOfBookings();
    ioSCTree.put ("bkgs", lNbOfBookings);
    // Put nb of group bookings in property tree
    const NbOfBookings_T& lNbOfGroupBookings =
      iBookingClass.getNbOfGroupBookings();
    ioSCTree.put ("grp_bks (pdg)", lNbOfGroupBookings);
    // Put nb of staff bookings in property tree
    const NbOfBookings_T& lNbOfStaffBookings =
      iBookingClass.getNbOfStaffBookings();
    ioSCTree.put ("stf_bkgs", lNbOfStaffBookings);
    // Put nb of WL bookings in property tree
    const NbOfBookings_T& lNbOfWLBookings =
      iBookingClass.getNbOfWLBookings();
    ioSCTree.put ("wl_bkgs", lNbOfWLBookings);
    // Put ETB in property tree
    const NbOfBookings_T& lETB = iBookingClass.getETB();
    ioSCTree.put ("etb", lETB);
    // Put net class availability in property tree
    const Availability_T& lNetClassAvailability =
      iBookingClass.getNetClassAvailability();
    ioSCTree.put ("class_avl", lNetClassAvailability);
    // Put segment availability in property tree
    const Availability_T& lSegmentAvailability =
      iBookingClass.getSegmentAvailability();
    ioSCTree.put ("seg_avl", lSegmentAvailability);
    // Put net revenue availability in property tree
    const Availability_T& lNetRevenueAvailability =
      iBookingClass.getNetRevenueAvailability();
    ioSCTree.put ("rev_avl", lNetRevenueAvailability);

    // Add the sub-classe (containing cabin and fare-families information)
    // to the global tree
    ioPropertyTree.push_back(std::make_pair("", ioSCTree));

#endif // BOOST_VERSION >= 104100
  }

  // ////////////////////////////////////////////////////////////////////
  void BomJSONExport::
  jsonExportBookingRequestObject (std::ostream& oStream,
                                  const EventStruct& iEventStruct) {

    // Get the current event type: it should be booking request
    const EventType::EN_EventType& lEventType =
      iEventStruct.getEventType();
    assert (lEventType == EventType::BKG_REQ);

    // Get the booking request (the current event type is booking request)
    const BookingRequestStruct& lBookingRequest =
      iEventStruct.getBookingRequest();

#if BOOST_VERSION >= 104100

    // Create an empty property tree object for the current booking request
    bpt::ptree ptBookingRequest;

    // Put request date time in property tree
    const DateTime_T& lRequestDateTime =
      lBookingRequest.getRequestDateTime();
    ptBookingRequest.put ("time_stamp", lRequestDateTime);
```

```
    // Put event type in property tree
    ptBookingRequest.put ("event_type", EventType::getLabel(lEventType));

    // Put origin in property tree
    const AirportCode_T& lOrigin = lBookingRequest.getOrigin();
    ptBookingRequest.put ("org", lOrigin);
    // Put destination in property tree
    const AirportCode_T& lDestination = lBookingRequest.getDestination();
    ptBookingRequest.put ("des", lDestination);
    // Put preferred cabin in property tree
    const CabinCode_T& lCabinCode = lBookingRequest.getPreferredCabin();
    ptBookingRequest.put ("cab", lCabinCode);
    // Put party size in property tree
    const NbOfSeats_T& lNbOfSeats = lBookingRequest.getPartySize();
    ptBookingRequest.put ("pax", lNbOfSeats);
    // Put point-of-sale in property tree
    const AirportCode_T& lPOS = lBookingRequest.getPOS();
    ptBookingRequest.put ("pos", lPOS);
    // Put channel in property tree
    const ChannelLabel_T& lChannelLabel =
      lBookingRequest.getBookingChannel();
    ptBookingRequest.put ("cha", lChannelLabel);
    // Put WTP in property tree
    const WTP_T& lWTP = lBookingRequest.getWTP();
    ptBookingRequest.put ("wtp", lWTP);
    // Put request date in property tree
    const Date_T& lRequestDate =
      lRequestDateTime.boost::posix_time::ptime::date();
    ptBookingRequest.put ("bkg_date", lRequestDate);
    // Put departure date in property tree
    const Date_T& lPreferedDepartureDate =
      lBookingRequest.getPreferedDepartureDate();
    ptBookingRequest.put ("dep_date", lPreferedDepartureDate);
    // Put advance purchase in property tree
    assert (lPreferedDepartureDate >= lRequestDate);
    const DateOffset_T& lAdvancePurchase =
      lPreferedDepartureDate - lRequestDate;
    ptBookingRequest.put ("adv_purchase", lAdvancePurchase);
    // Put stay duration in property tree
    const DayDuration_T& lStayDuration =
      lBookingRequest.getStayDuration();
    ptBookingRequest.put ("stay_duration", lStayDuration);
    //  Put return date in property tree
    const DateOffset_T lDayDuration (lStayDuration);
    const Date_T& lReturnDate =
      lPreferedDepartureDate + lDayDuration;
    ptBookingRequest.put ("return_date", lReturnDate);
    // Put cancellation date in property tree
    // TODO:  cancellation date
    ptBookingRequest.put ("cancel_date", "xxxx-xx-xx");
    // Put preferred departure time in property tree
    const Duration_T& lPreferredDepartureTime =
      lBookingRequest.getPreferredDepartureTime();
    ptBookingRequest.put ("dep_time", lPreferredDepartureTime);
    // Put preferred return time in property tree
    // TODO: preferred return time
    ptBookingRequest.put ("return_time", "xxPM");
    // Put preferred carriers in property tree
    // TODO: preferred carriers
    ptBookingRequest.put ("pref_carriers", "XX");

    // Write the property tree into the JSON stream.
    write_json (oStream, ptBookingRequest);

#endif // BOOST_VERSION >= 104100
  }
```

```
00747   // ////////////////////////////////////////////////////////////////////
00748   void BomJSONExport::
00749   jsonExportBreakPointObject (std::ostream& oStream,
00750                               const EventStruct& iEventStruct) {
00751
00752     // Get the current event type: it should be break point
00753     const EventType::EN_EventType& lEventType =
00754       iEventStruct.getEventType();
00755     assert (lEventType == EventType::BRK_PT);
00756
00757     // Get the break point (the current event type is break point)
00758     const BreakPointStruct& lBreakPoint =
00759       iEventStruct.getBreakPoint();
00760
00761 #if BOOST_VERSION >= 104100
00762
00763     // Create an empty property tree object for the current break point
00764     bpt::ptree ptBreakPoint;
00765
00766     // Put break point date time in property tree
00767     const DateTime_T& lRequestDateTime =
00768       lBreakPoint.getBreakPointTime();
00769     ptBreakPoint.put ("time_stamp", lRequestDateTime);
00770     // Put event type in property tree
00771     ptBreakPoint.put ("event_type", EventType::getLabel(lEventType));

00772
00773     // Write the property tree into the JSON stream.
00774     write_json (oStream, ptBreakPoint);
00775
00776
00777 #endif // BOOST_VERSION >= 104100
00778   }
00779
00780 }
```

33.199 stdair/bom/BomJSONExport.hpp File Reference

```
#include <iosfwd>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Classes

- class stdair::BomJSONExport

  Utility class to export StdAir objects in a JSON format.

Namespaces

- namespace bpt
- namespace stdair

  Handle on the StdAir library context.

33.200 stdair/bom/BomJSONExport.hpp

```
#ifndef __STDAIR_BOM_BOMJSONEXPORT_HPP
#define __STDAIR_BOM_BOMJSONEXPORT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// Boost Property Tree
#if BOOST_VERSION >= 104100
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/bom/TravelSolutionTypes.hpp>

#if BOOST_VERSION >= 104100
  namespace bpt = boost::property_tree;
#else // BOOST_VERSION >= 104100
  namespace bpt {
    typedef char ptree;
  }
#endif // BOOST_VERSION >= 104100

namespace stdair {

  class BomRoot;
  class Inventory;
  class FlightDate;
  class LegDate;
  class LegCabin;
  class SegmentDate;
  class SegmentCabin;
  class BookingClass;
  class EventStruct;

  class BomJSONExport {
  public:
    // /////////////// Export support methods /////////////////

    static void jsonExportFlightDateList (std::ostream&, const BomRoot&,
                                          const AirlineCode_T& iAirlineCode = "all",
                                          const FlightNumber_T& iFlightNumber = 0);

    static void jsonExportFlightDateObjects (std::ostream&, const FlightDate&);

    static void jsonExportBookingRequestObject (std::ostream&,
                                                const EventStruct&);

    static void jsonExportBreakPointObject (std::ostream&,
                                            const EventStruct&);

  private:

    static void jsonExportFlightDate (bpt::ptree&,
                                      const Inventory&,
                                      const FlightNumber_T&);

    static void jsonExportLegDate (bpt::ptree&, const FlightDate&);

    static void jsonExportLegCabin (bpt::ptree&, const LegDate&);

    static void jsonExportBucket (bpt::ptree&, const LegCabin&);
```

```
00146
00156     static void jsonExportSegmentDate (bpt::ptree&, const FlightDate&);
00157
00166     static void jsonExportSegmentCabin (bpt::ptree&, const SegmentDate&);
00167
00180     static void jsonExportFareFamily (bpt::ptree&, bpt::ptree&,
00181                                       const SegmentCabin&);
00182
00192     static void jsonExportBookingClass (bpt::ptree&, bpt::ptree&,
00193                                         const BookingClass&);
00194
00195   };
00196
00197 }
00198 #endif // __STDAIR_BOM_BOMJSONEXPORT_HPP
```

33.201 stdair/bom/BomJSONImport.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

  Handle on the StdAir library context.

33.202 stdair/bom/BomJSONImport.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
#if BOOST_VERSION >= 104100
// Boost Property Tree
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/regex.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>

#if BOOST_VERSION >= 104100
namespace bpt = boost::property_tree;
#else // BOOST_VERSION >= 104100
namespace bpt {
  typedef char ptree;
}
#endif // BOOST_VERSION >= 104100

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::
  jsonImportCommand (const JSONString& iBomJSONStr,
                     JSonCommand::EN_JSonCommand& ioEnumJSonCommand) {

    bool hasCommandBeenSuccessfullyRetrieved = true;

    try {
      const std::string lRegEx("^[{][[:space:]]*\""
                               "([[:alpha:]|_]*)\"[[:space:]]*:"
                               "[[]?"
                               "[[:space:]]*[{]?"
                               "([[:alnum:]|[:punct:]|[:space:]]*)"
                               "[}]?[]]?[}]");

      // See the caller for the regular expression
      boost::regex lExpression (lRegEx);

      const std::string& lBomJSONStr = iBomJSONStr.getString();
      std::string::const_iterator itStart = lBomJSONStr.begin();
      std::string::const_iterator itEnd = lBomJSONStr.end();

      boost::match_results<std::string::const_iterator> lWhat;
      boost::match_flag_type lFlags = boost::match_default;

      regex_search (itStart, itEnd, lWhat, lExpression, lFlags);

      // Put the matched strings in the list of tokens to be returned back
      // to the caller
      std::vector<std::string> oTokenList;
      for (boost::match_results<std::string::const_iterator>::const_iterator itMatch
            = lWhat.begin(); itMatch != lWhat.end(); ++itMatch) {

        const std::string lMatchedString (std::string (itMatch->first,
                                                       itMatch->second));
```

```
        oTokenList.push_back (lMatchedString);
      }

      // If the retrieved token list is empty, the command has not been
      // retrieved
      if (oTokenList.size() <= 1) {
        hasCommandBeenSuccessfullyRetrieved = false;
        return hasCommandBeenSuccessfullyRetrieved;
      }

      assert (oTokenList.size() >= 2);
      // Retrieved the command string into the token list
      const std::string lCommandStr = oTokenList.at(1);
      const JSonCommand lJSonCommand (lCommandStr);
      ioEnumJSonCommand = lJSonCommand.getCommand();

    } catch (stdair::CodeConversionException& ccException) {
      hasCommandBeenSuccessfullyRetrieved = false;
    }

    return hasCommandBeenSuccessfullyRetrieved;

  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportInventoryKey (const JSONString& iBomJSONStr,
                                              AirlineCode_T& ioAirlineCode) {
    bool hasKeyBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Build the right path to obtain the airline code value.
      bpt::ptree::const_iterator itBegin = pt.begin();
      const std::string lCommandName = itBegin->first;
      std::ostringstream lPath;
      lPath << lCommandName << ".airline_code";

      // Get the airline_code.
      // If the path key is not found, an exception is thrown.
      ioAirlineCode = pt.get<AirlineCode_T> (lPath.str());

    } catch (bpt::ptree_error& bptException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    }

#endif // BOOST_VERSION >= 104100
    return hasKeyBeenSuccessfullyRetrieved;
  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportFlightDate (const JSONString& iBomJSONStr,
                                            Date_T& ioDepartureDate) {
    bool hasKeyBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;
```

```

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Build the right path to obtain the departure date value.
      const std::string& lDepartureDateStr =
        pt.get<std::string> ("flight_date.departure_date");

      // Get the departure_date.
      // If the path key is not found, an exception is thrown.
      ioDepartureDate =
        boost::gregorian::from_simple_string (lDepartureDateStr);

    } catch (bpt::ptree_error& bptException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasKeyBeenSuccessfullyRetrieved;
  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportFlightNumber (const JSONString& iBomJSONStr,
                                              FlightNumber_T& ioFlightNumber) {

    bool hasKeyBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Build the right path to obtain the flight number value.
      bpt::ptree::const_iterator itBegin = pt.begin();
      const std::string lCommandName = itBegin->first;
      std::ostringstream lPath;
      lPath << lCommandName << ".flight_number";

      // Get the flight_number.
      // If the path key is not found, an exception is thrown.
      ioFlightNumber = pt.get<FlightNumber_T> (lPath.str());

    } catch (bpt::ptree_error& bptException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasKeyBeenSuccessfullyRetrieved;
  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportBreakPoints (const JSONString& iBomJSONStr,
                                             BreakPointList_T& oBreakPointList) {

```

```
    bool hasKeyBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Access the break point list tree
      bpt::ptree::const_iterator itBegin = pt.begin();
      bpt::ptree ptListOfBP = itBegin->second;
      // Browse the break point list
      for (bpt::ptree::const_iterator itBP = ptListOfBP.begin();
           itBP != ptListOfBP.end(); ++itBP) {
        // Access the current break point tree
        bpt::ptree ptBP = itBP->second;
        // Access the date of the break point
        bpt::ptree::const_iterator itDate = ptBP.begin();
        bpt::ptree ptDate = itDate->second;
        // Recover the string containing the date
        std::string lDateString = ptDate.data();
        if (lDateString.empty() == false) {
          // Construct the break point using the recovered string
          const Date_T lDate =
            boost::gregorian::from_simple_string (lDateString);
          BreakPointStruct lBreakPoint (lDate);
          // Add the break point to the list
          oBreakPointList.push_back (lBreakPoint);
        }
      }
    } catch (bpt::ptree_error& bptException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    } catch (boost::bad_lexical_cast& eCast) {
      hasKeyBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasKeyBeenSuccessfullyRetrieved;
  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportEventType (const JSONString& iBomJSONStr,
                                           EventType::EN_EventType& ioEventType)
  {

    bool hasKeyBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Build the right path to obtain the event type value.
      bpt::ptree::const_iterator itBegin = pt.begin();
```

```
      const std::string lEventTypeName = itBegin->first;
      std::ostringstream lPath;
      lPath << lEventTypeName << ".event_type";

      // Get the event type string
      // If the path key is not found, an exception bpt::ptree_error is thrown.
      const std::string lEventTypeStr = pt.get<std::string> (lPath.str());
      // Build the event type using the string.
      // If the input string is incorrect, an exception
      // stdair::CodeConversionException is thrown.
      const EventType lEventType (lEventTypeStr);
      ioEventType = lEventType.getType();

    } catch (bpt::ptree_error& bptException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    } catch (stdair::CodeConversionException& cceException) {
      hasKeyBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasKeyBeenSuccessfullyRetrieved;
  }

  // ////////////////////////////////////////////////////////////////////
  bool BomJSONImport::jsonImportConfig (const JSONString& iBomJSONStr,
                                        ConfigHolderStruct& iConfigHolderStruct)
  {

    bool hasConfigBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    // Create an empty property tree object
    bpt::ptree pt;

    try {

      // Load the JSON formatted string into the property tree.
      // If reading fails (cannot open stream, parse error), an
      // exception is thrown.
      std::istringstream iStr (iBomJSONStr.getString());
      read_json (iStr, pt);

      // Load the pt in the configuration holder
      iConfigHolderStruct.add (pt);
    } catch (bpt::ptree_error& bptException) {
      hasConfigBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasConfigBeenSuccessfullyRetrieved;
  }

}
```

33.203 stdair/bom/BomJSONImport.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
```

Classes

- class stdair::BomJSONImport

  Utility class to import StdAir objects in a JSON format.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.204 stdair/bom/BomJSONImport.hpp

```
#ifndef __STDAIR_BOM_BOMJSONIMPORT_HPP
#define __STDAIR_BOM_BOMJSONIMPORT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/JSonCommand.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/BreakPointStruct.hpp>


namespace stdair {

  class JSONString;
  class ConfigHolderStruct;

  class BomJSONImport {
  public:
    // ///////////////// Import support methods /////////////////
    static bool jsonImportCommand (const JSONString&,
                                   JSonCommand::EN_JSonCommand&);
    static bool jsonImportInventoryKey (const JSONString&,
                                        AirlineCode_T&);

    static bool jsonImportFlightDate (const JSONString&,
                                      Date_T&);

    static bool jsonImportFlightNumber (const JSONString&,
                                        FlightNumber_T&);

    static bool jsonImportBreakPoints (const JSONString&,
                                       BreakPointList_T&);

    static bool jsonImportEventType (const JSONString&,
                                     EventType::EN_EventType&);

    static bool jsonImportConfig (const JSONString&,
                                  ConfigHolderStruct&);
  };

}
#endif // __STDAIR_BOM_BOMJSONIMPORT_HPP
```

33.205 stdair/bom/BomKeyManager.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::tokenizer< boost::char_separator< char > > stdair::Tokeniser_T

33.206 stdair/bom/BomKeyManager.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_BomDisplay.hpp>
00014 #include <stdair/bom/InventoryKey.hpp>
00015 #include <stdair/bom/FlightDateKey.hpp>
00016 #include <stdair/bom/SegmentDateKey.hpp>
00017 #include <stdair/bom/LegDateKey.hpp>
00018 #include <stdair/bom/ParsedKey.hpp>
00019 #include <stdair/bom/BomKeyManager.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024   // ////////////// Tokenising support ///////////////
00028   typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   ParsedKey BomKeyManager::extractKeys (const std::string& iFullKeyStr) {
00032     ParsedKey oParsedKey;
00033     oParsedKey._fullKey = iFullKeyStr;
00034
00035     // Token-ise the full key string
00036     Tokeniser_T lTokens (iFullKeyStr, DEFAULT_KEY_TOKEN_DELIMITER);
00037     Tokeniser_T::iterator itToken = lTokens.begin();
00038
00039     // Airline code
00040     if (itToken != lTokens.end()) {
00041       oParsedKey._airlineCode = *itToken;
00042
00043     // Flight number
00044       ++itToken;
00045       if (itToken != lTokens.end()) {
00046         oParsedKey._flightNumber = *itToken;
00047
00048         // Departure date
00049         ++itToken;
00050         if (itToken != lTokens.end()) {
00051           oParsedKey._departureDate = *itToken;
00052
00053           // Origin
00054           ++itToken;
00055           if (itToken != lTokens.end()) {
00056             oParsedKey._boardingPoint = *itToken;
00057
00058             // Destination
00059             ++itToken;
00060             if (itToken != lTokens.end()) {
00061               oParsedKey._offPoint = *itToken;
00062
00063               // Boarding time
00064               ++itToken;
00065               if (itToken != lTokens.end()) {
00066                 oParsedKey._boardingTime = *itToken;
00067               }
00068             }
```

```
            }
          }
        }
      }

      return oParsedKey;
    }

    // ////////////////////////////////////////////////////////////////////
    InventoryKey BomKeyManager::
    extractInventoryKey (const std::string& iFullKeyStr) {
      ParsedKey lParsedKey = extractKeys (iFullKeyStr);

      return lParsedKey.getInventoryKey();
    }

    // ////////////////////////////////////////////////////////////////////
    FlightDateKey BomKeyManager::
    extractFlightDateKey (const std::string& iFullKeyStr) {
      ParsedKey lParsedKey = extractKeys (iFullKeyStr);

      return lParsedKey.getFlightDateKey();
    }

    // ////////////////////////////////////////////////////////////////////
    SegmentDateKey BomKeyManager::
    extractSegmentDateKey (const std::string& iFullKeyStr) {
      ParsedKey lParsedKey = extractKeys (iFullKeyStr);

      return lParsedKey.getSegmentKey();
    }

    // ////////////////////////////////////////////////////////////////////
    LegDateKey BomKeyManager::
    extractLegDateKey (const std::string& iFullKeyStr) {
      ParsedKey lParsedKey = extractKeys (iFullKeyStr);

      return lParsedKey.getLegKey();
    }
}
```

33.207 stdair/bom/BomKeyManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class stdair::BomKeyManager

  Utility class to extract key structures from strings.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.208 stdair/bom/BomKeyManager.hpp

```
#ifndef __STDAIR_BOM_BOMKEYMANAGER_HPP
#define __STDAIR_BOM_BOMKEYMANAGER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// StdAir
#include <stdair/stdair_basic_types.hpp>

namespace stdair {

  struct BomRootKey;
  struct InventoryKey;
  struct FlightDateKey;
  struct LegDateKey;
  struct SegmentDateKey;
  struct LegCabinKey;
  struct SegmentCabinKey;
  struct FareFamilyKey;
  struct BookingClassKey;
  struct ParsedKey;

  class BomKeyManager {
  public:
    // ///////////////// Key management support methods /////////////////
    static ParsedKey extractKeys (const std::string& iFullKeyStr);

    static InventoryKey extractInventoryKey (const std::string& iFullKeyStr);

    static FlightDateKey extractFlightDateKey (const std::string& iFullKeyStr);

    static SegmentDateKey extractSegmentDateKey (const std::string& iFullKeyStr);

    static LegDateKey extractLegDateKey (const std::string& iFullKeyStr);

  };

}
#endif // __STDAIR_BOM_BOMKEYMANAGER_HPP
```

33.209 stdair/bom/BomManager.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
```

Classes

- class stdair::BomManager

  Utility class for StdAir-based objects.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.210 stdair/bom/BomManager.hpp

```
#ifndef __STDAIR_BOM_BOMMANAGER_HPP
#define __STDAIR_BOM_BOMMANAGER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
#include <list>
#include <map>
// Boost
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/service/Logger.hpp>
// Stdair BOM Objects
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>

namespace stdair {

  class BomManager {
    friend class FacBomManager;

  public:
    template <typename OBJECT2, typename OBJECT1>
    static const typename BomHolder<OBJECT2>::BomList_T& getList(const OBJECT1&);

    template <typename OBJECT2, typename OBJECT1>
    static const typename BomHolder<OBJECT2>::BomMap_T& getMap (const OBJECT1&);

    template <typename OBJECT2, typename OBJECT1>
    static bool hasList (const OBJECT1&);

    template <typename OBJECT2, typename OBJECT1>
    static bool hasMap (const OBJECT1&);

    template <typename PARENT, typename CHILD>
    static PARENT* getParentPtr (const CHILD&);

    template <typename PARENT, typename CHILD>
    static PARENT& getParent (const CHILD&);

    template <typename OBJECT2, typename OBJECT1>
    static OBJECT2* getObjectPtr (const OBJECT1&, const MapKey_T&);

    template <typename OBJECT2, typename OBJECT1>
    static OBJECT2& getObject (const OBJECT1&, const MapKey_T&);


  private:
    template <typename OBJECT2, typename OBJECT1>
    static const BomHolder<OBJECT2>& getBomHolder (const OBJECT1&);
  };

  // ////////////////////////////////////////////////////////////////////
  // Private method.
  template <typename OBJECT2, typename OBJECT1>
  const BomHolder<OBJECT2>& BomManager::getBomHolder (const OBJECT1& iObject1) {
```

```

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    const HolderMap_T& lHolderMap = iObject1.getHolderMap();

    HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));

    if (itHolder == lHolderMap.end()) {
      const std::string lName (typeid (OBJECT2).name());
      throw NonInitialisedContainerException ("Cannot find the holder of type "
                                              + lName + " within: "
                                              + iObject1.describeKey());
    }

    const BomHolder<OBJECT2>* lBomHolder_ptr =
      static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
    assert (lBomHolder_ptr != NULL);

    return *lBomHolder_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // This method is specialized for the following couple types:
  // <SegmentDate, SegmentDate>
  template <typename OBJECT2, typename OBJECT1>
  const typename BomHolder<OBJECT2>::BomList_T& BomManager::
  getList (const OBJECT1& iObject1) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

  const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
    return lBomHolder._bomList;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT2, typename OBJECT1>
  const typename BomHolder<OBJECT2>::BomMap_T& BomManager::
  getMap (const OBJECT1& iObject1) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
```

```
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                          || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    const BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (iObject1);
    return lBomHolder._bomMap;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // This method is specialized for the following couple types:
  // <SegmentDate, SegmentDate>
  template <typename OBJECT2, typename OBJECT1>
  bool BomManager::hasList (const OBJECT1& iObject1) {

    const HolderMap_T& lHolderMap = iObject1.getHolderMap();
    HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));

    if (itHolder == lHolderMap.end()) {
      return false;
    }
    const BomHolder<OBJECT2>* lBomHolder_ptr =
      static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
    assert (lBomHolder_ptr != NULL);

    return !lBomHolder_ptr->_bomList.empty();
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // This method is specialized for the following couple types:
  // <SegmentDate, SegmentDate>
  template <typename OBJECT2, typename OBJECT1>
  bool BomManager::hasMap (const OBJECT1& iObject1) {

    const HolderMap_T& lHolderMap = iObject1.getHolderMap();
    HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (OBJECT2));

    if (itHolder == lHolderMap.end()) {
      return false;
    }
    const BomHolder<OBJECT2>* lBomHolder_ptr =
      static_cast<const BomHolder<OBJECT2>*> (itHolder->second);
    assert (lBomHolder_ptr != NULL);

    return !lBomHolder_ptr->_bomMap.empty();
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method valid for all PARENT and CHILD types.
  // (No compile time assertion to check PARENT and CHILD types.)
  template <typename PARENT, typename CHILD>
  PARENT* BomManager::getParentPtr (const CHILD& iChild) {

    PARENT* const lParent_ptr = static_cast<PARENT* const> (iChild.getParent());
    return lParent_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method valid for all PARENT and CHILD types.
  // (No compile time assertion to check PARENT and CHILD types.)
  template <typename PARENT, typename CHILD>
  PARENT& BomManager::getParent (const CHILD& iChild) {

    PARENT* const lParent_ptr = getParentPtr<PARENT> (iChild);
```

```
    assert (lParent_ptr != NULL);
    return *lParent_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT2, typename OBJECT1>
  OBJECT2* BomManager::getObjectPtr (const OBJECT1& iObject1,
                                     const MapKey_T& iKey) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));

    OBJECT2* oBom_ptr = NULL;

    const HolderMap_T& lHolderMap = iObject1.getHolderMap();

    typename HolderMap_T::const_iterator itHolder =
      lHolderMap.find (&typeid (OBJECT2));

    if (itHolder != lHolderMap.end()) {

      BomHolder<OBJECT2>* const lBomHolder_ptr =
        static_cast<BomHolder<OBJECT2>* const> (itHolder->second);
      assert (lBomHolder_ptr != NULL);

      //
      typedef typename BomHolder<OBJECT2>::BomMap_T BomMap_T;
      BomMap_T& lBomMap =  lBomHolder_ptr->_bomMap;
      typename BomMap_T::iterator itBom = lBomMap.find (iKey);

      if (itBom != lBomMap.end()) {
        oBom_ptr = itBom->second;
        assert (oBom_ptr != NULL);
      }
    }

    return oBom_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT2, typename OBJECT1>
  OBJECT2& BomManager::getObject (const OBJECT1& iObject1,
                                  const MapKey_T& iKey) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));

    OBJECT2* oBom_ptr = NULL;

    typedef std::map<const MapKey_T, OBJECT2*> BomMap_T;
    const BomMap_T& lBomMap = getMap<OBJECT2> (iObject1);
```

```

  typename BomMap_T::const_iterator itBom = lBomMap.find (iKey);

  if (itBom == lBomMap.end()) {
    const std::string lName (typeid (OBJECT2).name());

    STDAIR_LOG_ERROR ("Cannot find the objet of type " << lName
                      << " with key " << iKey << " within: "
                      << iObject1.describeKey());
    assert (false);
  }

  oBom_ptr = itBom->second;
  assert (oBom_ptr != NULL);

  return *oBom_ptr;
}

// ////////////////////////////////////////////////////////////////////
//
// Specialization of the template methods above for a segment
// date and its corresponding marketing segment dates.
//
// ////////////////////////////////////////////////////////////////////

// Specialization of the template method hasList above for the types
// <SegmentDate, SegmentDate>.
// Return a boolean saying if the marketing segment date list is empty
// or not.
template<>
inline bool BomManager::hasList<SegmentDate,SegmentDate>
(const SegmentDate& ioSegmentDate) {

  const SegmentDateList_T& lMarketingSegmentDateList =
    ioSegmentDate.getMarketingSegmentDateList ();
  const bool isMarketingSegmentDateListEmpty =
    lMarketingSegmentDateList.empty();
  const bool hasMarketingSegmentDateList =
    !isMarketingSegmentDateListEmpty;
  return hasMarketingSegmentDateList;
}

// Specialization of the template method hasList above for the types
// <SegmentDate, SegmentDate>.
// Return the marketing segment date list.
template<>
inline const BomHolder<SegmentDate>::BomList_T&
BomManager::getList<SegmentDate,SegmentDate> (const SegmentDate& ioSegmentDate)
{

  const SegmentDateList_T& lMarketingSegmentDateList =
    ioSegmentDate.getMarketingSegmentDateList ();
  return lMarketingSegmentDateList;
}

// Specialization of the template method hasMap above for the types
// <SegmentDate, SegmentDate>.
// A segment date does not have a Segment Date Map but it can have a
// Segment Date list (containing its marketing segment dates).
template<>
inline bool BomManager::hasMap<SegmentDate,SegmentDate>
(const SegmentDate& ioSegmentDate) {

  const bool hasMap = false;
  return hasMap;
}

```

```
00364   // ////////////////////////////////////////////////////////////////////
00365   //
00366   // Specialization of the template methods above for an inventory
00367   // and its airline features.
00368   //
00369   // ////////////////////////////////////////////////////////////////////
00370
00371   // Specialization of the template method hasList above for the types
00372   // <AirlineFeature,Inventory>.
00373   template<>
00374   inline bool BomManager::hasList<AirlineFeature,Inventory>
00375   (const Inventory& ioInventory) {
00376
00377     const bool hasList = false;
00378     return hasList;
00379   }
00380
00381   // Specialization of the template method hasMap above for the types
00382   // <AirlineFeature,Inventory>.
00383   template<>
00384   inline bool BomManager::hasMap<AirlineFeature,Inventory>
00385   (const Inventory& ioInventory) {
00386
00387     const bool hasMap = false;
00388     return hasMap;
00389   }
00390
00391   // Specialization of the template method getObjectPtr above for the types
00392   // <AirlineFeature,Inventory>.
00393   template<>
00394   inline AirlineFeature* BomManager::getObjectPtr<AirlineFeature,Inventory>
00395   (const Inventory& iInventory, const MapKey_T& iKey) {
00396
00397     AirlineFeature* lAirlineFeature_ptr = iInventory.getAirlineFeature ();
00398
00399     return lAirlineFeature_ptr;
00400   }
00401
00402   // Specialization of the template method getObject above for the types
00403   // <AirlineFeature,Inventory>.
00404   template<>
00405   inline AirlineFeature& BomManager::getObject<AirlineFeature,Inventory>
00406   (const Inventory& iInventory, const MapKey_T& iKey) {
00407
00408     AirlineFeature* lAirlineFeature_ptr =
00409       getObjectPtr<AirlineFeature,Inventory> (iInventory, iKey);
00410     assert (lAirlineFeature_ptr != NULL);
00411
00412     return *lAirlineFeature_ptr;
00413   }
00414
00415
00416 }
00417 #endif // __STDAIR_BOM_BOMMANAGER_HPP
```

33.211 stdair/bom/BomRetriever.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.212 stdair/bom/BomRetriever.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/basic/BasConst_BomDisplay.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/AirlineFeature.hpp>
00015 #include <stdair/bom/FlightDate.hpp>
00016 #include <stdair/bom/LegDate.hpp>
00017 #include <stdair/bom/SegmentDate.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 #include <stdair/bom/SegmentCabin.hpp>
00020 #include <stdair/bom/FareFamily.hpp>
00021 #include <stdair/bom/BookingClass.hpp>
00022 #include <stdair/bom/BomRetriever.hpp>
00023 #include <stdair/bom/ParsedKey.hpp>
00024 #include <stdair/bom/AirportPair.hpp>
00025 #include <stdair/service/Logger.hpp>
00026
00027 namespace stdair {
00028
00029   // ////////////////////////////////////////////////////////////////////
00030   Inventory* BomRetriever::
00031   retrieveInventoryFromLongKey (const BomRoot& iBomRoot,
00032                                 const std::string& iFullKeyStr) {
00033     Inventory* oInventory_ptr = NULL;
00034
00035     // Extract the inventory key (i.e., airline code)
00036     const InventoryKey& lInventoryKey =
00037       BomKeyManager::extractInventoryKey (iFullKeyStr);
00038
00039     oInventory_ptr = iBomRoot.getInventory (lInventoryKey);
00040
00041     return oInventory_ptr;
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   Inventory* BomRetriever::
00046   retrieveInventoryFromLongKey (const Inventory& iInventory,
00047                                 const std::string& iFullKeyStr) {
00048     Inventory* oInventory_ptr = NULL;
00049
00050     // Extract the inventory key (i.e., airline code)
00051     const InventoryKey& lInventoryKey =
00052       BomKeyManager::extractInventoryKey (iFullKeyStr);
00053     const stdair::AirlineCode_T lAirlineCode =
00054       lInventoryKey.getAirlineCode();
00055
00056     oInventory_ptr =
00057       BomManager::getObjectPtr<Inventory> (iInventory,
00058                                            lAirlineCode);
00059     return oInventory_ptr;
00060   }
00061
00062   // ////////////////////////////////////////////////////////////////////
00063   Inventory* BomRetriever::retrieveInventoryFromKey (const BomRoot& iBomRoot,
00064                                                      const InventoryKey& iKey) {
00065     Inventory* oInventory_ptr = NULL;
```

```

  //
  oInventory_ptr = iBomRoot.getInventory (iKey);

  return oInventory_ptr;
}

// ////////////////////////////////////////////////////////////////////
Inventory* BomRetriever::
retrieveInventoryFromKey (const BomRoot& iBomRoot,
                          const AirlineCode_T& iAirlineCode) {
  Inventory* oInventory_ptr = NULL;

  //
  const InventoryKey lKey (iAirlineCode);
  oInventory_ptr = iBomRoot.getInventory (lKey);

  return oInventory_ptr;
}

// ////////////////////////////////////////////////////////////////////
AirlineFeature* BomRetriever::
retrieveAirlineFeatureFromKey (const BomRoot& iBomRoot,
                               const AirlineCode_T& iAirlineCode) {
  Inventory* oInventory_ptr = NULL;
  AirlineFeature* oAirlineFeature_ptr = NULL;

  //
  oInventory_ptr = retrieveInventoryFromKey (iBomRoot, iAirlineCode);
  if (oInventory_ptr == NULL) {
    return oAirlineFeature_ptr;
  }
  assert (oInventory_ptr != NULL);

  oAirlineFeature_ptr =
    BomManager::getObjectPtr<AirlineFeature,Inventory> (*oInventory_ptr,
                                                        iAirlineCode);

  return oAirlineFeature_ptr;
}

// ////////////////////////////////////////////////////////////////////
FlightDate* BomRetriever::
retrieveFlightDateFromLongKey (const BomRoot& iBomRoot,
                               const std::string& iFullKeyStr) {
  FlightDate* oFlightDate_ptr = NULL;

  // Retrieve the inventory
  Inventory* oInventory_ptr =
    BomRetriever::retrieveInventoryFromLongKey (iBomRoot, iFullKeyStr);
  if (oInventory_ptr == NULL) {
    return oFlightDate_ptr;
  }
  assert (oInventory_ptr != NULL);

  // Extract the flight-date key (i.e., flight number and date)
  const FlightDateKey& lFlightDateKey =
    BomKeyManager::extractFlightDateKey (iFullKeyStr);

  oFlightDate_ptr = oInventory_ptr->getFlightDate (lFlightDateKey);

  return oFlightDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
FlightDate* BomRetriever::
retrieveFlightDateFromKeySet (const BomRoot& iBomRoot,
```

```
                                const AirlineCode_T& iAirlineCode,
                                const FlightNumber_T& iFlightNumber,
                                const Date_T& iFlightDateDate) {
  FlightDate* oFlightDate_ptr = NULL;

  // Retrieve the inventory
  Inventory* oInventory_ptr =
    BomRetriever::retrieveInventoryFromKey (iBomRoot, iAirlineCode);
  if (oInventory_ptr == NULL) {
    return oFlightDate_ptr;
  }
  assert (oInventory_ptr != NULL);

  //
  oFlightDate_ptr = retrieveFlightDateFromKey (*oInventory_ptr,
                                               iFlightNumber, iFlightDateDate);

  return oFlightDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
FlightDate* BomRetriever::
retrieveFlightDateFromLongKey (const Inventory& iInventory,
                               const std::string& iFullKeyStr) {
  FlightDate* oFlightDate_ptr = NULL;

  // Extract the flight-date key (i.e., flight number and date)
  const FlightDateKey& lFlightDateKey =
    BomKeyManager::extractFlightDateKey (iFullKeyStr);

  oFlightDate_ptr = iInventory.getFlightDate (lFlightDateKey);

  return oFlightDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
FlightDate* BomRetriever::
retrieveFlightDateFromKey (const Inventory& iInventory,
                           const FlightDateKey& iKey) {
  FlightDate* oFlightDate_ptr = NULL;

  //
  oFlightDate_ptr = iInventory.getFlightDate (iKey);

  return oFlightDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
FlightDate* BomRetriever::
retrieveFlightDateFromKey (const Inventory& iInventory,
                           const FlightNumber_T& iFlightNumber,
                           const Date_T& iFlightDateDate) {
  FlightDate* oFlightDate_ptr = NULL;

  //
  const FlightDateKey lKey (iFlightNumber, iFlightDateDate);
  oFlightDate_ptr = iInventory.getFlightDate (lKey);

  return oFlightDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrieveSegmentDateFromLongKey (const BomRoot& iBomRoot,
                                const std::string& iFullKeyStr) {
  SegmentDate* oSegmentDate_ptr = NULL;
```

```

  // Retrieve the flight-date
  FlightDate* oFlightDate_ptr =
    BomRetriever::retrieveFlightDateFromLongKey (iBomRoot, iFullKeyStr);
  if (oFlightDate_ptr == NULL) {
    return oSegmentDate_ptr;
  }
  assert (oFlightDate_ptr != NULL);

  // Extract the segment-date key (i.e., origin and destination)
  const SegmentDateKey& lSegmentDateKey =
    BomKeyManager::extractSegmentDateKey (iFullKeyStr);

  oSegmentDate_ptr = oFlightDate_ptr->getSegmentDate (lSegmentDateKey);

  return oSegmentDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrieveSegmentDateFromLongKey (const Inventory& iInventory,
                                const std::string& iFullKeyStr) {
  SegmentDate* oSegmentDate_ptr = NULL;

  ParsedKey lParsedKey = BomKeyManager::extractKeys (iFullKeyStr);

  if (iInventory.getAirlineCode() != lParsedKey._airlineCode) {
    STDAIR_LOG_DEBUG ("Airline code: " << lParsedKey._airlineCode);
    return oSegmentDate_ptr;
  }

  FlightDate* lFlightDate_ptr =
    retrieveFlightDateFromKey (iInventory, lParsedKey.getFlightDateKey());
  if (lFlightDate_ptr == NULL) {
    STDAIR_LOG_DEBUG ("Flight-date key: "
                      << lParsedKey.getFlightDateKey().toString());
    return oSegmentDate_ptr;
  }

  oSegmentDate_ptr =
    retrieveSegmentDateFromKey (*lFlightDate_ptr, lParsedKey.getSegmentKey());
  if (oSegmentDate_ptr == NULL) {
    STDAIR_LOG_DEBUG ("Segment-date key: "
                      << lParsedKey.getSegmentKey().toString());
    return oSegmentDate_ptr;
  }

  return oSegmentDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrieveSegmentDateFromLongKey (const FlightDate& iFlightDate,
                                const std::string& iFullKeyStr) {
  SegmentDate* oSegmentDate_ptr = NULL;

  // Extract the segment-date key (i.e., origin and destination)
  const SegmentDateKey& lSegmentDateKey =
    BomKeyManager::extractSegmentDateKey (iFullKeyStr);

  oSegmentDate_ptr = iFlightDate.getSegmentDate (lSegmentDateKey);

  return oSegmentDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
LegDate* BomRetriever::
```

```
retrieveOperatingLegDateFromLongKey (const FlightDate& iFlightDate,
                                     const std::string& iFullKeyStr) {
  LegDate* oLegDate_ptr = NULL;

  // Extract the segment-date key (i.e., origin and destination)
  const LegDateKey& lLegDateKey =
    BomKeyManager::extractLegDateKey (iFullKeyStr);

  oLegDate_ptr = iFlightDate.getLegDate (lLegDateKey);

  return oLegDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrievePartnerSegmentDateFromLongKey (const Inventory& iInventory,
                                       const std::string& iFullKeyStr) {
  SegmentDate* oSegmentDate_ptr = NULL;
  Inventory* oInventory_ptr = NULL;

  // Extract the inventory key (i.e., airline code)
  const InventoryKey& lInventoryKey =
    BomKeyManager::extractInventoryKey (iFullKeyStr);
  const stdair::AirlineCode_T lAirlineCode =
    lInventoryKey.getAirlineCode();

  // Retrieve the inventory
  oInventory_ptr =
    retrieveInventoryFromLongKey (iInventory, lAirlineCode);

  if (oInventory_ptr != NULL) {
    oSegmentDate_ptr =
      retrieveSegmentDateFromLongKey (*oInventory_ptr, iFullKeyStr);
  }

  return oSegmentDate_ptr;

}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
                            const SegmentDateKey& iKey) {
  SegmentDate* oSegmentDate_ptr = NULL;

  //
  oSegmentDate_ptr = iFlightDate.getSegmentDate (iKey);

  return oSegmentDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate* BomRetriever::
retrieveSegmentDateFromKey (const FlightDate& iFlightDate,
                            const AirportCode_T& iOrigin,
                            const AirportCode_T& iDestination) {
  SegmentDate* oSegmentDate_ptr = NULL;

  //
  const SegmentDateKey lKey (iOrigin, iDestination);
  oSegmentDate_ptr = iFlightDate.getSegmentDate (lKey);

  return oSegmentDate_ptr;
}

// ////////////////////////////////////////////////////////////////////
BookingClass* BomRetriever::
```

```
retrieveBookingClassFromLongKey (const Inventory& iInventory,
                                 const std::string& iFullKeyStr,
                                 const ClassCode_T& iClassCode) {
  BookingClass* oBookingClass_ptr = NULL;

  SegmentDate* lSegmentDate_ptr = retrieveSegmentDateFromLongKey (iInventory,
                                                                  iFullKeyStr);

  if (lSegmentDate_ptr == NULL) {
    return oBookingClass_ptr;
  }
  assert (lSegmentDate_ptr != NULL);

  //
  oBookingClass_ptr =
    BomManager::getObjectPtr<BookingClass> (*lSegmentDate_ptr, iClassCode);

  return oBookingClass_ptr;
}

// ////////////////////////////////////////////////////////////////////
AirportPair* BomRetriever::
retrieveAirportPairFromKeySet (const BomRoot& iBomRoot,
                               const stdair::AirportCode_T& iOrigin,
                               const stdair::AirportCode_T& iDestination) {

  // Get the Airport pair stream of the segment path.
  const AirportPairKey lAirportPairKey (iOrigin, iDestination);

  // Search for the fare rules having the same origin and
  // destination airport as the travel solution
  AirportPair* oAirportPair_ptr = BomManager::
    getObjectPtr<AirportPair> (iBomRoot, lAirportPairKey.toString());

  return oAirportPair_ptr;

}

// ////////////////////////////////////////////////////////////////////
void BomRetriever::
retrieveDatePeriodListFromKey (const AirportPair& iAirportPair,
                               const stdair::Date_T& iDepartureDate,
                               stdair::DatePeriodList_T& ioDatePeriodList) {

  // Get the list of date-period
  const DatePeriodList_T& lFareDatePeriodList =
    BomManager::getList<DatePeriod> (iAirportPair);

  // Browse the date-period list
  for (DatePeriodList_T::const_iterator itDateRange =
         lFareDatePeriodList.begin();
       itDateRange != lFareDatePeriodList.end(); ++itDateRange) {

    DatePeriod* lCurrentFareDatePeriod_ptr = *itDateRange ;
    assert (lCurrentFareDatePeriod_ptr != NULL);

    // Select the date-period objects having a corresponding date range
    const bool isDepartureDateValid =
      lCurrentFareDatePeriod_ptr->isDepartureDateValid (iDepartureDate);

    // Add the date-period objects having a corresponding date range
    // to the list to display
    if (isDepartureDateValid == true) {
      ioDatePeriodList.push_back(lCurrentFareDatePeriod_ptr);
    }
  }
```

```
00399
00400   }
00401
00402   // ////////////////////////////////////////////////////////////////////
00403   void BomRetriever::
00404   retrieveDatePeriodListFromKeySet (const BomRoot& iBomRoot,
00405                                     const stdair::AirportCode_T& iOrigin,
00406                                     const stdair::AirportCode_T& iDestination,
00407                                     const stdair::Date_T& iDepartureDate,
00408                                     stdair::DatePeriodList_T& ioDatePeriodList) {

00409
00410     // Retrieve the airport-pair
00411     AirportPair* oAirportPair_ptr =
00412       BomRetriever::retrieveAirportPairFromKeySet(iBomRoot, iOrigin,
00413                                                   iDestination);
00414     if (oAirportPair_ptr == NULL) {
00415       return;
00416     }
00417     assert (oAirportPair_ptr != NULL);
00418
00419     // Retrieve the flight date
00420     BomRetriever::retrieveDatePeriodListFromKey (*oAirportPair_ptr, iDepartureDat
      e,
00421                                                  ioDatePeriodList);
00422
00423   }
00424
00425   // ////////////////////////////////////////////////////////////////////
00426   LegCabin& BomRetriever::
00427   retrieveDummyLegCabin (stdair::BomRoot& iBomRoot,
00428                          const bool isForFareFamilies) {
00429
00430     LegCabin* oLegCabin_ptr = NULL;
00431
00432     // Retrieve the Inventory
00433     const Inventory* lInventory_ptr = BomRetriever::
00434       retrieveInventoryFromKey (iBomRoot, DEFAULT_AIRLINE_CODE);
00435
00436     if (lInventory_ptr == NULL) {
00437       std::ostringstream oStr;
00438       oStr << "The inventory corresponding to the '"
00439            << DEFAULT_AIRLINE_CODE << "' airline can not be found";
00440       throw ObjectNotFoundException (oStr.str());
00441     }
00442
00443     // Retrieve the FlightDate
00444     FlightDate* lFlightDate_ptr = NULL;
00445     if (isForFareFamilies == true) {
00446       lFlightDate_ptr = BomRetriever::
00447         retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER_FF,
00448                                    DEFAULT_DEPARTURE_DATE);
00449
00450       if (lFlightDate_ptr == NULL) {
00451         std::ostringstream oStr;
00452         oStr << "The flight-date corresponding to ("
00453              << DEFAULT_FLIGHT_NUMBER_FF << ", "
00454              << DEFAULT_DEPARTURE_DATE << ") can not be found";
00455         throw ObjectNotFoundException (oStr.str());
00456       }
00457     } else {
00458       lFlightDate_ptr = BomRetriever::
00459         retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER,
00460                                    DEFAULT_DEPARTURE_DATE);
00461
00462       if (lFlightDate_ptr == NULL) {
00463         std::ostringstream oStr;
```

```
        oStr << "The flight-date corresponding to ("
             << DEFAULT_FLIGHT_NUMBER << ", "
             << DEFAULT_DEPARTURE_DATE << ") can not be found";
        throw ObjectNotFoundException (oStr.str());
      }
    }
    assert(lFlightDate_ptr != NULL);

    // Retrieve the LegDate
    const LegDateKey lLegDateKey (DEFAULT_ORIGIN);
    const LegDate* lLegDate_ptr =
      lFlightDate_ptr->getLegDate (lLegDateKey);

    if (lLegDate_ptr == NULL) {
      std::ostringstream oStr;
      oStr << "The leg-date corresponding to the '"
           << DEFAULT_ORIGIN << "' origin can not be found";
      throw ObjectNotFoundException (oStr.str());
    }

    // Retrieve the LegCabin
    const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
    oLegCabin_ptr = lLegDate_ptr->getLegCabin (lLegCabinKey);

    if (oLegCabin_ptr == NULL) {
      std::ostringstream oStr;
      oStr << "The leg-cabin corresponding to the '"
           << DEFAULT_CABIN_CODE << "' cabin code can not be found";
      throw ObjectNotFoundException (oStr.str());
    }

    assert (oLegCabin_ptr != NULL);
    return *oLegCabin_ptr;

  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabin& BomRetriever::
  retrieveDummySegmentCabin (stdair::BomRoot& iBomRoot,
                             const bool isForFareFamilies) {

    SegmentCabin* oSegmentCabin_ptr = NULL;

    // Retrieve the Inventory
    const Inventory* lInventory_ptr = BomRetriever::
      retrieveInventoryFromKey (iBomRoot, DEFAULT_AIRLINE_CODE);

    if (lInventory_ptr == NULL) {
      std::ostringstream oStr;
      oStr << "The inventory corresponding to the '"
           << DEFAULT_AIRLINE_CODE << "' airline can not be found";
      throw ObjectNotFoundException (oStr.str());
    }

    // Retrieve the FlightDate
    FlightDate* lFlightDate_ptr = NULL;
    if (isForFareFamilies == true) {
      lFlightDate_ptr = BomRetriever::
        retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER_FF,
                                   DEFAULT_DEPARTURE_DATE);

      if (lFlightDate_ptr == NULL) {
        std::ostringstream oStr;
        oStr << "The flight-date corresponding to ("
             << DEFAULT_FLIGHT_NUMBER_FF << ", "
             << DEFAULT_DEPARTURE_DATE << ") can not be found";
        throw ObjectNotFoundException (oStr.str());
```

```
      }
    } else {
      lFlightDate_ptr = BomRetriever::
        retrieveFlightDateFromKey (*lInventory_ptr, DEFAULT_FLIGHT_NUMBER,
                                   DEFAULT_DEPARTURE_DATE);

      if (lFlightDate_ptr == NULL) {
        std::ostringstream oStr;
        oStr << "The flight-date corresponding to ("
             << DEFAULT_FLIGHT_NUMBER << ", "
             << DEFAULT_DEPARTURE_DATE << ") can not be found";
        throw ObjectNotFoundException (oStr.str());
      }
    }
    assert(lFlightDate_ptr != NULL);

    // Retrieve the SegmentDate
    const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
    const SegmentDate* lSegmentDate_ptr =
      lFlightDate_ptr->getSegmentDate (lSegmentDateKey);

    if (lSegmentDate_ptr == NULL) {
      std::ostringstream oStr;
      oStr << "The segment-date corresponding to the '"
           << DEFAULT_ORIGIN << "' origin and '"
           << DEFAULT_DESTINATION << "' destination can not be found";
      throw ObjectNotFoundException (oStr.str());
    }

    // Retrieve the SegmentCabin
    const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
    oSegmentCabin_ptr =
      BomManager::getObjectPtr<SegmentCabin> (*lSegmentDate_ptr, lSegmentCabinKey
.toString());

    if (oSegmentCabin_ptr == NULL) {
      std::ostringstream oStr;
      oStr << "The segment-cabin corresponding to the '"
           << DEFAULT_CABIN_CODE << "' cabin code can not be found";
      throw ObjectNotFoundException (oStr.str());
    }

    assert (oSegmentCabin_ptr != NULL);
    return *oSegmentCabin_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BomRetriever::
  retrieveFullKeyFromSegmentDate (const SegmentDate& iSegmentdate) {

    std::ostringstream lFullKeyStr;

    // Get the parent flight date
    FlightDate* lFlightDate_ptr =
      BomManager::getParentPtr<FlightDate>(iSegmentdate);
    if (lFlightDate_ptr == NULL) {
      return lFullKeyStr.str();
    }
    assert (lFlightDate_ptr != NULL);

    // Get the parent inventory
    Inventory* lInventory_ptr =
      BomManager::getParentPtr<Inventory> (*lFlightDate_ptr);
    if (lInventory_ptr == NULL) {
      return lFullKeyStr.str();
    }
    assert (lInventory_ptr != NULL);
```

```

    lFullKeyStr << lInventory_ptr->describeKey()
                << DEFAULT_KEY_SUB_FLD_DELIMITER;
    lFullKeyStr << lFlightDate_ptr->describeKey()
                << DEFAULT_KEY_SUB_FLD_DELIMITER;
    lFullKeyStr << iSegmentdate.describeKey();

    return lFullKeyStr.str();

  }

}
```

33.213 stdair/bom/BomRetriever.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Classes

- class stdair::BomRetriever

  Utility class to retrieve StdAir objects.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.214 stdair/bom/BomRetriever.hpp

```
#ifndef __STDAIR_BOM_BOMRETRIEVER_HPP
#define __STDAIR_BOM_BOMRETRIEVER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/DatePeriod.hpp>

namespace stdair {

  class BomRoot;
  struct InventoryKey;
  class Inventory;
  class AirlineFeature;
  struct FlightDateKey;
  class FlightDate;
  class LegDate;
  struct SegmentDateKey;
  class SegmentDate;
  class LegCabin;
  class SegmentCabin;
  class FareFamily;
  class BookingClass;
  class DatePeriod;
  class AirportPair;

  class BomRetriever {
  public:
    // ///////////////// Key management support methods /////////////////
    static Inventory*
    retrieveInventoryFromLongKey (const BomRoot&,
                                  const std::string& iFullKeyStr);

    static Inventory*
    retrieveInventoryFromLongKey (const Inventory&,
                                  const std::string& iFullKeyStr);

    static Inventory* retrieveInventoryFromKey (const BomRoot&,
                                                const InventoryKey&);

    static Inventory* retrieveInventoryFromKey (const BomRoot&,
                                                const AirlineCode_T&);

    static AirlineFeature* retrieveAirlineFeatureFromKey (const BomRoot&,
                                                          const AirlineCode_T&);

    static FlightDate*
    retrieveFlightDateFromLongKey (const BomRoot&,
                                   const std::string& iFullKeyStr);

    static FlightDate*
    retrieveFlightDateFromKeySet (const BomRoot&,
                                  const AirlineCode_T&, const FlightNumber_T&,
                                  const Date_T& iFlightDateDate);

    static FlightDate*
    retrieveFlightDateFromLongKey (const Inventory&,
                                   const std::string& iFullKeyStr);

    static FlightDate* retrieveFlightDateFromKey (const Inventory&,
```

```
                                                const FlightDateKey&);

    static FlightDate* retrieveFlightDateFromKey (const Inventory&,
                                                const FlightNumber_T&,
                                                const Date_T& iFlightDateDate);


    static LegDate*
    retrieveOperatingLegDateFromLongKey (const FlightDate&,
                                         const std::string& iFullKeyStr);

    static SegmentDate*
    retrievePartnerSegmentDateFromLongKey (const Inventory&,
                                           const std::string& iFullKeyStr);

    static SegmentDate*
    retrieveSegmentDateFromLongKey (const BomRoot&,
                                    const std::string& iFullKeyStr);

    static SegmentDate*
    retrieveSegmentDateFromLongKey (const Inventory&,
                                    const std::string& iFullKeyStr);

    static SegmentDate*
    retrieveSegmentDateFromLongKey (const FlightDate&,
                                    const std::string& iFullKeyStr);

    static SegmentDate* retrieveSegmentDateFromKey (const FlightDate&,
                                                    const SegmentDateKey&);

    static SegmentDate*
    retrieveSegmentDateFromKey (const FlightDate&,
                                const AirportCode_T& iOrigin,
                                const AirportCode_T& iDestination);

    static BookingClass*
    retrieveBookingClassFromLongKey (const Inventory&,
                                     const std::string& iFullKeyStr,
                                     const ClassCode_T&);


    static AirportPair*
    retrieveAirportPairFromKeySet (const BomRoot& ,
                                   const stdair::AirportCode_T&,
                                   const stdair::AirportCode_T&);

    static void
    retrieveDatePeriodListFromKey (const AirportPair&,
                                   const stdair::Date_T&,
                                   stdair::DatePeriodList_T&);

    static void
    retrieveDatePeriodListFromKeySet (const BomRoot&,
                                      const stdair::AirportCode_T&,
                                      const stdair::AirportCode_T&,
                                      const stdair::Date_T&,
                                      stdair::DatePeriodList_T&);

    static stdair::LegCabin&
    retrieveDummyLegCabin (stdair::BomRoot&,
                           const bool isForFareFamilies = false);

    static stdair::SegmentCabin&
    retrieveDummySegmentCabin (stdair::BomRoot&,
                               const bool isForFareFamilies = false);

```

```
    static std::string retrieveFullKeyFromSegmentDate (const SegmentDate&);

  };

}
#endif // __STDAIR_BOM_BOMRETRIEVER_HPP
```

33.215 stdair/bom/BomRoot.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/Inventory.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.216 stdair/bom/BomRoot.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/Inventory.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BomRoot::BomRoot() {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot::BomRoot (const BomRoot& iBomRoot) :
    _key (iBomRoot._key), _frat5CurveHolder (iBomRoot._frat5CurveHolder),
    _ffDisutilityCurveHolder (iBomRoot._ffDisutilityCurveHolder) {
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot::BomRoot (const Key_T& iKey) : _key (iKey) {
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot::~BomRoot() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string BomRoot::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  Inventory* BomRoot::getInventory (const std::string& iInventoryKeyStr) const {
    Inventory* oInventory_ptr =
      BomManager::getObjectPtr<Inventory> (*this, iInventoryKeyStr);
    return oInventory_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  Inventory* BomRoot::getInventory (const InventoryKey& iInventoryKey) const {
    return getInventory (iInventoryKey.toString());
  }

}
```

33.217 stdair/bom/BomRoot.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomRootKey.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
```

Classes

- class stdair::BomRoot

  Class representing the actual attributes for the Bom root.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.218 stdair/bom/BomRoot.hpp

```
#ifndef __STDAIR_BOM_BOMROOT_HPP
#define __STDAIR_BOM_BOMROOT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BomRootKey.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct InventoryKey;
  class Inventory;

  class BomRoot : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    typedef BomRootKey Key_T;


  public:
    // ////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const FRAT5Curve_T& getFRAT5Curve (const std::string& iKey) const {
      return _frat5CurveHolder.getFRAT5Curve (iKey);
    }

    const FFDisutilityCurve_T& getFFDisutilityCurve (const std::string& iKey) const{
      return _ffDisutilityCurveHolder.getFFDisutilityCurve (iKey);
    }

    Inventory* getInventory (const std::string& iInventoryKeyStr) const;

    Inventory* getInventory (const InventoryKey&) const;

    // ////////////// Business Methods //////////
    void addFRAT5Curve (const std::string& iKey, const FRAT5Curve_T& iCurve) {
      _frat5CurveHolder.addCurve (iKey, iCurve);
    }

    void addFFDisutilityCurve (const std::string& iKey,
```

```
                                  const FFDisutilityCurve_T& iCurve) {
      _ffDisutilityCurveHolder.addCurve (iKey, iCurve);
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  protected:
    // ////////// Constructors and destructors /////////
    BomRoot();

    BomRoot (const BomRoot&);

    BomRoot (const Key_T& iKey);

    ~BomRoot();


  protected:
    // /////////////// Attributes ////////////////
    Key_T _key;

    HolderMap_T _holderMap;

    FRAT5CurveHolderStruct _frat5CurveHolder;

    FFDisutilityCurveHolderStruct _ffDisutilityCurveHolder;
  };

}
#endif // __STDAIR_BOM_BOMROOT_HPP
```

33.219 stdair/bom/BomRootKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRootKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::BomRootKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BomRootKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.220 stdair/bom/BomRootKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/bom/BomRootKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   BomRootKey::BomRootKey()
00019     : _id (DEFAULT_BOM_ROOT_KEY) {
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   BomRootKey::BomRootKey (const BomRootKey& iBomRootKey)
00024     : _id (iBomRootKey._id) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   BomRootKey::BomRootKey (const std::string& iIdentification)
00029     : _id (iIdentification) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   BomRootKey::~BomRootKey() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void BomRootKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "BomRootKey: " << toString() << std::endl;
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void BomRootKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string BomRootKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _id;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void BomRootKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void BomRootKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void BomRootKey::serialize (Archive& ioArchive,
00069                               const unsigned int iFileVersion) {
00070     ioArchive & _id;
00071   }
00072
00073   // ////////////////////////////////////////////////////////////////////
00074   // Explicit template instantiation
00075   namespace ba = boost::archive;
00076   template void BomRootKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                            unsigned int);
00078   template void BomRootKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                            unsigned int);
00080   // ////////////////////////////////////////////////////////////////////
00081
00082 }
```

33.221 stdair/bom/BomRootKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::BomRootKey

  Key of the BOM structure root.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.222 stdair/bom/BomRootKey.hpp

```
#ifndef __STDAIR_BOM_BOMROOTKEY_HPP
#define __STDAIR_BOM_BOMROOTKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/bom/KeyAbstract.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct BomRootKey : public KeyAbstract {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  public:
    BomRootKey ();

    BomRootKey (const std::string& iIdentification);

    BomRootKey (const BomRootKey&);

    ~BomRootKey();


  public:
    // /////////// Getters //////////
    const std::string& getID() const {
      return _id;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  private:
    // ///////////////// Attributes ///////////////
    std::string _id;
  };

```

```
00114 }
00115 #endif // __STDAIR_BOM_BOMROOTKEY_HPP
```

33.223 stdair/bom/BookingClass.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/RandomGeneration.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.224 stdair/bom/BookingClass.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/RandomGeneration.hpp>
00011 #include <stdair/bom/BookingClass.hpp>
00012
00013 namespace stdair {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   BookingClass::BookingClass() : _key (DEFAULT_CLASS_CODE), _parent (NULL) {
00017     assert (false);
00018   }
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   BookingClass::BookingClass (const BookingClass& iBookingClass)
00022     : _key (iBookingClass._key),
00023       _parent (NULL),
00024       _subclassCode (iBookingClass._subclassCode),
00025       _cumulatedProtection (iBookingClass._cumulatedProtection),
00026       _protection (iBookingClass._protection),
00027       _cumulatedBookingLimit (iBookingClass._cumulatedBookingLimit),
00028       _au (iBookingClass._au),
00029       _nego (iBookingClass._nego),
00030       _noShowPercentage (iBookingClass._noShowPercentage),
00031       _cancellationPercentage (iBookingClass._cancellationPercentage),
00032       _nbOfBookings (iBookingClass._nbOfBookings),
00033       _groupNbOfBookings (iBookingClass._groupNbOfBookings),
00034       _groupPendingNbOfBookings (iBookingClass._groupPendingNbOfBookings),
00035       _staffNbOfBookings (iBookingClass._staffNbOfBookings),
00036       _wlNbOfBookings (iBookingClass._wlNbOfBookings),
00037       _nbOfCancellations (iBookingClass._nbOfCancellations),
00038       _etb (iBookingClass._etb),
00039       _netClassAvailability (iBookingClass._netClassAvailability),
00040       _segmentAvailability (iBookingClass._segmentAvailability),
00041       _netRevenueAvailability (iBookingClass._netRevenueAvailability),
00042       _yield (iBookingClass._yield),
00043       _adjustedYield (iBookingClass._adjustedYield),
00044       _mean (iBookingClass._mean),
00045       _stdDev (iBookingClass._stdDev) {
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   BookingClass::BookingClass (const Key_T& iKey)
00050     : _key (iKey), _parent (NULL), _subclassCode(0), _cumulatedProtection (0.0),
00051       _protection (0.0), _cumulatedBookingLimit (0.0), _au (0.0), _nego (0.0),
00052       _noShowPercentage (0.0), _cancellationPercentage (0.0),
00053       _nbOfBookings (0.0), _groupNbOfBookings (0.0),
00054       _groupPendingNbOfBookings (0.0), _staffNbOfBookings (0.0),
00055       _wlNbOfBookings (0.0), _nbOfCancellations (0.), _etb (0.0),
00056       _netClassAvailability (0.0), _segmentAvailability (0.0),
00057       _netRevenueAvailability (0.0), _yield (0.0), _mean (0.0), _stdDev (0.0) {
00058   }
00059
00060   // ////////////////////////////////////////////////////////////////////
00061   BookingClass::~BookingClass() {
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   std::string BookingClass::toString() const {
```

```
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClass::sell (const NbOfBookings_T& iNbOfBookings) {
    _nbOfBookings += iNbOfBookings;
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClass::cancel (const NbOfBookings_T& iNbOfCancellations) {
    _nbOfBookings -= iNbOfCancellations;
    _nbOfCancellations += iNbOfCancellations;
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClass::generateDemandSamples (const NbOfSamples_T& K) {
    _generatedDemandVector.clear();
    if (_stdDev > 0) {
      RandomGeneration lGenerator (DEFAULT_RANDOM_SEED);
      for (unsigned int i = 0; i < K; ++i) {
        RealNumber_T lDemandSample = lGenerator.generateNormal (_mean, _stdDev);
        _generatedDemandVector.push_back (lDemandSample);
      }
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClass::generateDemandSamples (const NbOfSamples_T& K,
                                            const RandomSeed_T& iSeed) {
    _generatedDemandVector.clear();
    if (_stdDev > 0) {
      RandomGeneration lGenerator (iSeed);
      for (unsigned int i = 0; i < K; ++i) {
        RealNumber_T lDemandSample = lGenerator.generateNormal (_mean, _stdDev);
        _generatedDemandVector.push_back (lDemandSample);
      }
    }
  }

}
```

33.225 stdair/bom/BookingClass.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassKey.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Classes

- class stdair::BookingClass

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.226 stdair/bom/BookingClass.hpp

```
#ifndef __STDAIR_BOM_BOOKINGCLASS_HPP
#define __STDAIR_BOM_BOOKINGCLASS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassKey.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

namespace stdair {

  class BookingClass : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;

  public:
    // ////////// Type definitions ////////////
    typedef BookingClassKey Key_T;

  public:
    // /////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    const ClassCode_T& getClassCode() const {
      return _key.getClassCode();
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const SubclassCode_T& getSubclassCode() const {
      return _subclassCode;
    }

    const AuthorizationLevel_T& getAuthorizationLevel() const {
      return _au;
    }

    const ProtectionLevel_T& getProtection() const {
      return _protection;
    }

    const ProtectionLevel_T& getCumulatedProtection() const {
      return _cumulatedProtection;
    }

    const BookingLimit_T& getCumulatedBookingLimit() const {
      return _cumulatedBookingLimit;
    }
```

```
00080
00082     const NbOfSeats_T& getNegotiatedSpace() const {
00083       return _nego;
00084     }
00085
00087     const OverbookingRate_T& getNoShowPercentage() const {
00088       return _noShowPercentage;
00089     }
00090
00092     const OverbookingRate_T& getCancellationPercentage() const {
00093       return _cancellationPercentage;
00094     }
00095
00097     const NbOfBookings_T& getNbOfBookings() const {
00098       return _nbOfBookings;
00099     }
00100
00102     const NbOfBookings_T& getNbOfGroupBookings() const {
00103       return _groupNbOfBookings;
00104     }
00105
00107     const NbOfBookings_T& getNbOfPendingGroupBookings() const {
00108       return _groupPendingNbOfBookings;
00109     }
00110
00112     const NbOfBookings_T& getNbOfStaffBookings() const {
00113       return _staffNbOfBookings;
00114     }
00115
00117     const NbOfBookings_T& getNbOfWLBookings() const {
00118       return _wlNbOfBookings;
00119     }
00120
00122     const NbOfCancellations_T& getNbOfCancellations() const {
00123       return _nbOfCancellations;
00124     }
00125
00127     const NbOfBookings_T& getETB() const {
00128       return _etb;
00129     }
00130
00132     const Availability_T& getNetClassAvailability() const {
00133       return _netClassAvailability;
00134     }
00135
00137     const Availability_T& getSegmentAvailability() const {
00138       return _segmentAvailability;
00139     }
00140
00142     const Availability_T& getNetRevenueAvailability() const {
00143       return _netRevenueAvailability;
00144     }
00145
00147     const Yield_T& getYield () const { return _yield; }
00148     const Yield_T& getAdjustedYield () const { return _adjustedYield; }
00149
00151     const MeanValue_T& getMean () const { return _mean; }
00152     const StdDevValue_T& getStdDev () const {return _stdDev; }
00153     const MeanValue_T& getPriceDemMean () const { return _priceDemMean; }
00154     const StdDevValue_T& getPriceDemStdDev () const { return _priceDemStdDev; }
00155     const MeanValue_T& getCumuPriceDemMean () const {
00156       return _cumuPriceDemMean;
00157     }
00158     const StdDevValue_T& getCumuPriceDemStdDev () const {
00159       return _cumuPriceDemStdDev;
00160     }
00161     const MeanValue_T& getProductDemMean () const { return _productDemMean; }
```

```
    const StdDevValue_T& getProductDemStdDev () const {return _productDemStdDev;
  }

    const GeneratedDemandVector_T& getGeneratedDemandVector () const {
      return _generatedDemandVector;
    }

  public:
    // /////////// Setters ////////////
    void setCumulatedProtection (const ProtectionLevel_T& iPL) {
      _cumulatedProtection = iPL;
    }

    void setProtection (const ProtectionLevel_T& iPL) {
      _protection = iPL;
    }

    void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
      _cumulatedBookingLimit = iBL;
    }

    void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
      _au = iAU;
    }

    void setSegmentAvailability (const Availability_T& iAvl) {
      _segmentAvailability = iAvl;
    }

    void setYield (const Yield_T& iYield) {
      _yield = iYield;
      _adjustedYield = iYield;
    }
    void setAdjustedYield (const Yield_T& iYield) { _adjustedYield = iYield; }

    void setMean (const MeanValue_T& iMean) { _mean = iMean; }
    void setStdDev (const StdDevValue_T& iStdDev) { _stdDev = iStdDev; }
    void setPriceDemMean (const MeanValue_T& iMean) { _priceDemMean = iMean; }
    void setPriceDemStdDev (const StdDevValue_T& iStdDev) {
      _priceDemStdDev = iStdDev;
    }
    void setCumuPriceDemMean (const MeanValue_T& iMean) {
      _cumuPriceDemMean = iMean; }
    void setCumuPriceDemStdDev (const StdDevValue_T& iStdDev) {
      _cumuPriceDemStdDev = iStdDev;
    }
    void setProductDemMean (const MeanValue_T& iMean) {
      _productDemMean = iMean;
    }
    void setProductDemStdDev (const StdDevValue_T& iStdDev) {
      _productDemStdDev = iStdDev;
    }

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }
```

```
00242
00243   public:
00244     // ////////////// Business Methods /////////////////
00246     void sell (const NbOfBookings_T&);
00247
00249     void cancel (const NbOfBookings_T&);
00250
00253     void generateDemandSamples (const NbOfSamples_T&);
00254
00257     void generateDemandSamples (const NbOfSamples_T&, const RandomSeed_T&);
00258
00259   protected:
00260     // ////////// Constructors and destructors /////////
00262     BookingClass (const Key_T&);
00264     virtual ~BookingClass();
00265
00266   private:
00268     BookingClass();
00270     BookingClass (const BookingClass&);
00271
00272
00273   protected:
00274     // ////////// Attributes /////////
00276     Key_T _key;
00277
00279     BomAbstract* _parent;
00280
00282     HolderMap_T _holderMap;
00283
00285     SubclassCode_T _subclassCode;
00286
00288     ProtectionLevel_T _cumulatedProtection;
00289
00291     ProtectionLevel_T _protection;
00292
00294     BookingLimit_T _cumulatedBookingLimit;
00295
00297     AuthorizationLevel_T _au;
00298
00300     NbOfSeats_T _nego;
00301
00303     OverbookingRate_T _noShowPercentage;
00304
00306     OverbookingRate_T _cancellationPercentage;
00307
00309     NbOfBookings_T _nbOfBookings;
00310
00312     NbOfBookings_T _groupNbOfBookings;
00313
00315     NbOfBookings_T _groupPendingNbOfBookings;
00316
00318     NbOfBookings_T _staffNbOfBookings;
00319
00321     NbOfBookings_T _wlNbOfBookings;
00322
00324     NbOfCancellations_T _nbOfCancellations;
00325
00327     NbOfBookings_T _etb;
00328
00330     Availability_T _netClassAvailability;
00331
00333     Availability_T _segmentAvailability;
00334
00336     Availability_T _netRevenueAvailability;
00337
00339     Yield_T _yield;
00340     Yield_T _adjustedYield;
```

```
00341
00343     MeanValue_T _mean;
00344     StdDevValue_T _stdDev;
00345
00347     MeanValue_T _priceDemMean;
00348     StdDevValue_T _priceDemStdDev;
00349
00351     MeanValue_T _cumuPriceDemMean;
00352     StdDevValue_T _cumuPriceDemStdDev;
00353
00355     MeanValue_T _productDemMean;
00356     StdDevValue_T _productDemStdDev;
00357
00359     GeneratedDemandVector_T _generatedDemandVector;
00360   };
00361
00362 }
00363 #endif // __STDAIR_BOM_BOOKINGCLASS_HPP
```

33.227 stdair/bom/BookingClassKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BookingClassKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.228 stdair/bom/BookingClassKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BookingClassKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BookingClassKey::BookingClassKey() : _classCode (DEFAULT_CLASS_CODE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  BookingClassKey::BookingClassKey (const BookingClassKey& iKey)
    : _classCode (iKey._classCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  BookingClassKey::BookingClassKey (const ClassCode_T& iClassCode)
    : _classCode (iClassCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  BookingClassKey::~BookingClassKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClassKey::toStream (std::ostream& ioOut) const {
    ioOut << "BookingClassKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void BookingClassKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BookingClassKey::toString() const {
    std::ostringstream oStr;
    oStr << _classCode;
    return oStr.str();
  }

}
```

33.229 stdair/bom/BookingClassKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::BookingClassKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.230 stdair/bom/BookingClassKey.hpp

```
00001 #ifndef __STDAIR_BOM_BOOKINGCLASSKEY_HPP
00002 #define __STDAIR_BOM_BOOKINGCLASSKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00016   struct BookingClassKey : public KeyAbstract {
00017
00018     // /////////// Constructors and destructors ///////////
00019   private:
00021     BookingClassKey();
00022
00023   public:
00025     BookingClassKey (const ClassCode_T& iClassCode);
00027     BookingClassKey (const BookingClassKey&);
00029     ~BookingClassKey();
00030
00031
00032     // /////////// Getters //////////
00034     const ClassCode_T& getClassCode () const {
00035       return _classCode;
00036     }
00037
00038
00039     // /////////// Display support methods /////////
00042     void toStream (std::ostream& ioOut) const;
00043
00046     void fromStream (std::istream& ioIn);
00047
00053     const std::string toString() const;
00054
00055
00056   private:
00057     // ///////////////// Attributes ///////////////
00059     ClassCode_T _classCode;
00060   };
00061
00062 }
00063 #endif // __STDAIR_BOM_BOOKINGCLASSKEY_HPP
```

33.231 stdair/bom/BookingClassTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< BookingClass * > stdair::BookingClassList_T
- typedef std::map< const MapKey_T, BookingClass * > stdair::BookingClassMap_T

33.232 stdair/bom/BookingClassTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_BOOKINGCLASSTYPES_HPP
#define __STDAIR_BOM_BOOKINGCLASSTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class BookingClass;

  typedef std::list<BookingClass*> BookingClassList_T;

  typedef std::map<const MapKey_T, BookingClass*> BookingClassMap_T;
}
#endif // __STDAIR_BOM_BOOKINGCLASSTYPES_HPP

```

33.233 stdair/bom/BookingRequestStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- void stdair::intDisplay (std::ostream &oStream, const int &iInt)

33.234 stdair/bom/BookingRequestStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/date_time/gregorian/formatters.hpp>
00009 #include <boost/date_time/posix_time/posix_time.hpp>
00010 // StdAir
00011 #include <stdair/basic/BasConst_Inventory.hpp>
00012 #include <stdair/basic/BasConst_Request.hpp>
00013 #include <stdair/bom/BookingRequestStruct.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   BookingRequestStruct::BookingRequestStruct()
00019     : _origin (DEFAULT_ORIGIN), _destination (DEFAULT_DESTINATION),
00020       _pos (DEFAULT_POS),
00021       _preferredDepartureDate (DEFAULT_PREFERRED_DEPARTURE_DATE),
00022       _preferredDepartureTime (DEFAULT_PREFERRED_DEPARTURE_TIME),
00023       _requestDateTime (DEFAULT_REQUEST_DATE_TIME),
00024       _preferredCabin (DEFAULT_PREFERRED_CABIN),
00025       _partySize (DEFAULT_PARTY_SIZE),
00026       _channel (DEFAULT_CHANNEL),
00027       _tripType (TRIP_TYPE_ONE_WAY),
00028       _stayDuration (DEFAULT_STAY_DURATION),
00029       _frequentFlyerType (DEFAULT_FF_TIER),
00030       _wtp (DEFAULT_WTP),
00031       _valueOfTime (DEFAULT_VALUE_OF_TIME),
00032       _changeFees (false), _changeFeeDisutility (0.0),
00033       _nonRefundable (false), _nonRefundableDisutility (0.0) {
00034     assert (false);
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   BookingRequestStruct::
00039   BookingRequestStruct (const BookingRequestStruct& iBookingRequest)
00040     : _generatorKey (iBookingRequest._generatorKey),
00041       _origin (iBookingRequest._origin),
00042       _destination (iBookingRequest._destination),
00043       _pos (iBookingRequest._pos),
00044       _preferredDepartureDate (iBookingRequest._preferredDepartureDate),
00045       _preferredDepartureTime (iBookingRequest._preferredDepartureTime),
00046       _requestDateTime (iBookingRequest._requestDateTime),
00047       _preferredCabin (iBookingRequest._preferredCabin),
00048       _partySize (iBookingRequest._partySize),
00049       _channel (iBookingRequest._channel),
00050       _tripType (iBookingRequest._tripType),
00051       _stayDuration (iBookingRequest._stayDuration),
00052       _frequentFlyerType (iBookingRequest._frequentFlyerType),
00053       _wtp (iBookingRequest._wtp),
00054       _valueOfTime (iBookingRequest._valueOfTime),
00055       _changeFees (iBookingRequest._changeFees),
00056       _changeFeeDisutility (iBookingRequest._changeFeeDisutility),
00057       _nonRefundable (iBookingRequest._nonRefundable),
00058       _nonRefundableDisutility (iBookingRequest._nonRefundableDisutility) {
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   BookingRequestStruct::
00063   BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey,
00064                         const AirportCode_T& iOrigin,
00065                         const AirportCode_T& iDestination,
```

```
                        const CityCode_T& iPOS,
                        const Date_T& iDepartureDate,
                        const DateTime_T& iRequestDateTime,
                        const CabinCode_T& iPreferredCabin,
                        const NbOfSeats_T& iPartySize,
                        const ChannelLabel_T& iChannel,
                        const TripType_T& iTripType,
                        const DayDuration_T& iStayDuration,
                        const FrequentFlyer_T& iFrequentFlyerType,
                        const Duration_T& iPreferredDepartureTime,
                        const WTP_T& iWTP,
                        const PriceValue_T& iValueOfTime,
                        const ChangeFees_T& iChangeFees,
                        const Disutility_T& iChangeFeeDisutility,
                        const NonRefundable_T& iNonRefundable,
                        const Disutility_T& iNonRefundableDisutility)
    : _generatorKey (iGeneratorKey), _origin (iOrigin),
      _destination (iDestination), _pos (iPOS),
      _preferredDepartureDate (iDepartureDate),
      _preferredDepartureTime (iPreferredDepartureTime),
      _requestDateTime (iRequestDateTime),
      _preferredCabin (iPreferredCabin), _partySize (iPartySize),
      _channel (iChannel), _tripType (iTripType),
      _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
      _wtp (iWTP), _valueOfTime (iValueOfTime),
      _changeFees (iChangeFees), _changeFeeDisutility (iChangeFeeDisutility),
      _nonRefundable (iNonRefundable),
      _nonRefundableDisutility (iNonRefundableDisutility) {
  }

  // ////////////////////////////////////////////////////////////////////
  BookingRequestStruct::
  BookingRequestStruct (const AirportCode_T& iOrigin,
                        const AirportCode_T& iDestination,
                        const CityCode_T& iPOS,
                        const Date_T& iDepartureDate,
                        const DateTime_T& iRequestDateTime,
                        const CabinCode_T& iPreferredCabin,
                        const NbOfSeats_T& iPartySize,
                        const ChannelLabel_T& iChannel,
                        const TripType_T& iTripType,
                        const DayDuration_T& iStayDuration,
                        const FrequentFlyer_T& iFrequentFlyerType,
                        const Duration_T& iPreferredDepartureTime,
                        const WTP_T& iWTP,
                        const PriceValue_T& iValueOfTime,
                        const ChangeFees_T& iChangeFees,
                        const Disutility_T& iChangeFeeDisutility,
                        const NonRefundable_T& iNonRefundable,
                        const Disutility_T& iNonRefundableDisutility)
    : _generatorKey (""), _origin (iOrigin),
      _destination (iDestination), _pos (iPOS),
      _preferredDepartureDate (iDepartureDate),
      _preferredDepartureTime (iPreferredDepartureTime),
      _requestDateTime (iRequestDateTime),
      _preferredCabin (iPreferredCabin), _partySize (iPartySize),
      _channel (iChannel), _tripType (iTripType),
      _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
      _wtp (iWTP), _valueOfTime (iValueOfTime),
      _changeFees (iChangeFees), _changeFeeDisutility (iChangeFeeDisutility),
      _nonRefundable (iNonRefundable),
      _nonRefundableDisutility (iNonRefundableDisutility) {
  }

  // ////////////////////////////////////////////////////////////////////
  BookingRequestStruct::~BookingRequestStruct() {
  }
```

```
// ////////////////////////////////////////////////////////////////////
void BookingRequestStruct::toStream (std::ostream& ioOut) const {
  ioOut << describe();
}

// ////////////////////////////////////////////////////////////////////
void BookingRequestStruct::fromStream (std::istream& ioIn) {
}

// ////////////////////////////////////////////////////////////////////
const std::string BookingRequestStruct::describe() const {
  std::ostringstream oStr;
  oStr << "At " << _requestDateTime
       << ", for (" << _pos << ", " << _channel << ")"
       << " " << _origin << "-" << _destination << " (" << _tripType << ")"
       << " " << _preferredDepartureDate << " (" << _stayDuration << " days)"
       << " " << _preferredDepartureTime
       << " " << _preferredCabin << " " << _partySize
       << " " << _frequentFlyerType << " " << _wtp << " " << _valueOfTime
       << " " << _changeFees << " " << _changeFeeDisutility << " "
       << _nonRefundable << " " << _nonRefundableDisutility;
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
void intDisplay (std::ostream& oStream, const int& iInt) {
  const int dInt = iInt - static_cast<int> (iInt / 100) * 100;
  if (dInt < 10) {
    oStream << "0" << dInt;
  } else {
    oStream << dInt;
  }
}

// ////////////////////////////////////////////////////////////////////
const std::string BookingRequestStruct::display() const {
  std::ostringstream oStr;

  // Request date and time
  const Date_T& lRequestDate = _requestDateTime.date();
  oStr << boost::gregorian::to_iso_extended_string (lRequestDate);

  const Duration_T& lRequestTime = _requestDateTime.time_of_day();
  oStr << ", " << boost::posix_time::to_simple_string (lRequestTime);

  // POS
  oStr << ", " << _pos;

  // Channel
  oStr << ", " << _channel;

  // Origin
  oStr << ", " << _origin;

  // Destination
  oStr << ", " << _destination;

  // Preferred departure date
  oStr << ", "
       << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);

  // Preferred departure time
  oStr << ", "
       << boost::posix_time::to_simple_string (_preferredDepartureTime);

  // MIN & MAX preferred departure time (hardcode)
```

```
    oStr << ", " << "00:00-23:59";

    // Preferred arrival date (hardcode to the preferred departure date)
    oStr << ", "
         << boost::gregorian::to_iso_extended_string (_preferredDepartureDate);

    // Preferred arrival time (hard-coded to 23:55)
    oStr << ", " << "23:55";

    // Preferred cabin
    oStr << ", " << _preferredCabin;

    // Trip type
    oStr << ", " << _tripType;

    // Duration of stay
    oStr << ", ";
    if (_tripType == TRIP_TYPE_ONE_WAY) {
      oStr << "0";
    } else {
      oStr << _stayDuration;
    }

    // Frequent flyer tier
    oStr << ", " << _frequentFlyerType;

    // Willingness-to-pay
    oStr << ", " << _wtp;

    // Disutility per stop (hardcode to 100, expressed as a monetary
    // unit per hour)
    oStr << ", " << "100";

    // Value of time
    oStr << ", " << _valueOfTime;

    // Change fees
    oStr << ", " << _changeFees;

    // Change fee disutility
    oStr << ", " << _changeFeeDisutility;

    // Non refundable
    oStr << ", " << _nonRefundable;

    // Non refundable disutility
    oStr << ", " << _nonRefundableDisutility;

    return oStr.str();
  }

}
```

33.235 stdair/bom/BookingRequestStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
```

Classes

- struct stdair::BookingRequestStruct

  Structure holding the elements of a booking request.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.236 stdair/bom/BookingRequestStruct.hpp

```
00001 #ifndef __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
00002 #define __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingRequestTypes.hpp>
00015
00016 namespace stdair {
00017
00021   struct BookingRequestStruct : public StructAbstract {
00022   public:
00023     // /////////////// Getters /////////////////
00025     const DemandGeneratorKey_T& getDemandGeneratorKey () const {
00026       return _generatorKey;
00027     }
00028
00030     const AirportCode_T& getOrigin() const {
00031       return _origin;
00032     }
00033
00035     const AirportCode_T& getDestination() const {
00036       return _destination;
00037     }
00038
00040     const CityCode_T& getPOS() const {
00041       return _pos;
00042     }
00043
00045     const Date_T& getPreferedDepartureDate() const {
00046       return _preferredDepartureDate;
00047     }
00048
00050     const Duration_T& getPreferredDepartureTime() const {
00051       return _preferredDepartureTime;
00052     }
00053
00055     const DateTime_T& getRequestDateTime() const {
00056       return _requestDateTime;
00057     }
00058
00060     const CabinCode_T& getPreferredCabin() const {
00061       return _preferredCabin;
00062     }
00063
00065     const NbOfSeats_T& getPartySize() const {
00066       return _partySize;
00067     }
00068
00070     const ChannelLabel_T& getBookingChannel() const {
00071       return _channel;
00072     }
00073
00075     const TripType_T& getTripType() const {
00076       return _tripType;
00077     }
00078
00080     const DayDuration_T& getStayDuration() const {
```

```
      return _stayDuration;
    }

    const FrequentFlyer_T& getFrequentFlyerType() const {
      return _frequentFlyerType;
    }

    const WTP_T& getWTP() const {
      return _wtp;
    }

    const PriceValue_T& getValueOfTime () const {
      return _valueOfTime;
    }

    const ChangeFees_T& getChangeFees () const {
      return _changeFees;
    }

    const Disutility_T& getChangeFeeDisutility () const {
      return _changeFeeDisutility;
    }

    const NonRefundable_T& getNonRefundable () const {
      return _nonRefundable;
    }

    const Disutility_T& getNonRefundableDisutility () const {
      return _nonRefundableDisutility;
    }


  public:
    // /////////// Display support method /////////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string describe() const;

    const std::string display() const;


  public:
    // /////////////// Constructors and Destructors /////////////////
    BookingRequestStruct (const DemandGeneratorKey_T& iGeneratorKey,
                          const AirportCode_T& iOrigin,
                          const AirportCode_T& iDestination,
                          const CityCode_T& iPOS,
                          const Date_T& iDepartureDate,
                          const DateTime_T& iRequestDateTime,
                          const CabinCode_T& iPreferredCabin,
                          const NbOfSeats_T& iPartySize,
                          const ChannelLabel_T& iChannel,
                          const TripType_T& iTripType,
                          const DayDuration_T& iStayDuration,
                          const FrequentFlyer_T& iFrequentFlyerType,
                          const Duration_T& iPreferredDepartureTime,
                          const WTP_T& iWTP,
                          const PriceValue_T& iValueOfTime,
                          const ChangeFees_T& iChangeFees,
                          const Disutility_T& iChangeFeeDisutility,
                          const NonRefundable_T& iNonRefundable,
                          const Disutility_T& iNonRefundableDisutility);

    BookingRequestStruct (const AirportCode_T& iOrigin,
                          const AirportCode_T& iDestination,
```

```
00222                           const CityCode_T& iPOS,
00223                           const Date_T& iDepartureDate,
00224                           const DateTime_T& iRequestDateTime,
00225                           const CabinCode_T& iPreferredCabin,
00226                           const NbOfSeats_T& iPartySize,
00227                           const ChannelLabel_T& iChannel,
00228                           const TripType_T& iTripType,
00229                           const DayDuration_T& iStayDuration,
00230                           const FrequentFlyer_T& iFrequentFlyerType,
00231                           const Duration_T& iPreferredDepartureTime,
00232                           const WTP_T& iWTP,
00233                           const PriceValue_T& iValueOfTime,
00234                           const ChangeFees_T& iChangeFees,
00235                           const Disutility_T& iChangeFeeDisutility,
00236                           const NonRefundable_T& iNonRefundable,
00237                           const Disutility_T& iNonRefundableDisutility);
00241     BookingRequestStruct (const BookingRequestStruct&);
00242
00246     ~BookingRequestStruct();
00247
00248
00249   private:
00255     BookingRequestStruct();
00256
00257
00258   private:
00259     // /////////////// Attributes /////////////////
00261     const DemandGeneratorKey_T _generatorKey;
00262
00264     const AirportCode_T _origin;
00265
00267     const AirportCode_T _destination;
00268
00270     const CityCode_T _pos;
00271
00273     const Date_T _preferredDepartureDate;
00274
00276     const Duration_T _preferredDepartureTime;
00277
00279     const DateTime_T _requestDateTime;
00280
00282     const CabinCode_T _preferredCabin;
00283
00285     const NbOfSeats_T _partySize;
00286
00288     const ChannelLabel_T _channel;
00289
00292     const TripType_T _tripType;
00293
00295     const DayDuration_T _stayDuration;
00296
00298     const FrequentFlyer_T _frequentFlyerType;
00299
00301     const WTP_T _wtp;
00302
00304     const PriceValue_T _valueOfTime;
00305
00307     const ChangeFees_T _changeFees;
00308
00310     const Disutility_T _changeFeeDisutility;
00311
00313     const NonRefundable_T _nonRefundable;
00314
00316     const Disutility_T _nonRefundableDisutility;
00317   };
00318
00319 }
```

```
#endif // __STDAIR_BOM_BOOKINGREQUESTSTRUCT_HPP
```

33.237 stdair/bom/BookingRequestTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< BookingRequestStruct > stdair::BookingRequestPtr_T
- typedef std::string stdair::DemandGeneratorKey_T

33.238 stdair/bom/BookingRequestTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00003 #define __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // Boost
00009 #include <boost/shared_ptr.hpp>
00010
00011 namespace stdair {
00012
00013   // Forward declarations
00014   struct BookingRequestStruct;
00015
00016   // ############# Type definitions ##############
00018   typedef boost::shared_ptr<BookingRequestStruct> BookingRequestPtr_T;
00019
00021   typedef std::string DemandGeneratorKey_T;
00022
00023
00024 }
00025 #endif // __STDAIR_BOM_BOOKINGREQUESTTYPES_HPP
00026
```

33.239 stdair/bom/BreakPointStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.240 stdair/bom/BreakPointStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BreakPointStruct.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BreakPointStruct::BreakPointStruct() {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  BreakPointStruct::
  BreakPointStruct (const BreakPointStruct& iBreakPoint)
    : _breakPointTime (iBreakPoint._breakPointTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  BreakPointStruct::
  BreakPointStruct (const DateTime_T& iBreakPointTime)
    : _breakPointTime (iBreakPointTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  BreakPointStruct::
  BreakPointStruct (const Date_T& iBreakPointDate)
    : _breakPointTime (iBreakPointDate, DEFAULT_NULL_DURATION) {
  }

  // ////////////////////////////////////////////////////////////////////
  BreakPointStruct::~BreakPointStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  void BreakPointStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void BreakPointStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BreakPointStruct::describe() const {
    std::ostringstream oStr;
    oStr << _breakPointTime;
    return oStr.str();
  }

}
```

33.241 stdair/bom/BreakPointStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BreakPointTypes.hpp>
```

Classes

- struct stdair::BreakPointStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.242 stdair/bom/BreakPointStruct.hpp

```
00001 #ifndef __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
00002 #define __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_date_time_types.hpp>
00012 #include <stdair/basic/StructAbstract.hpp>
00013 #include <stdair/bom/BreakPointTypes.hpp>
00014
00015 namespace stdair {
00016
00018   struct BreakPointStruct : public StructAbstract {
00019   public:
00020     // /////////////// Getters /////////////////
00022     const DateTime_T& getBreakPointTime() const {
00023       return _breakPointTime;
00024     }
00025
00026     // /////////// Display support method /////////////
00029     void toStream (std::ostream& ioOut) const;
00030
00033     void fromStream (std::istream& ioIn);
00034
00036     const std::string describe() const;
00037
00038
00039     // /////////////// Constructors and Destructors /////////////////
00040   public:
00042     BreakPointStruct (const DateTime_T&);
00043
00045     BreakPointStruct (const Date_T&);
00046
00048     BreakPointStruct (const BreakPointStruct&);
00049
00050   private:
00053     BreakPointStruct ();
00054
00055   public:
00057     ~BreakPointStruct();
00058
00059
00060   private:
00061     // /////////////// Attributes /////////////////
00063     const DateTime_T _breakPointTime;
00064   };
00065
00066 }
00067 #endif // __STDAIR_BOM_BREAKPOINTSTRUCT_HPP
```

33.243 stdair/bom/BreakPointTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< BreakPointStruct > stdair::BreakPointPtr_T
- typedef std::list< BreakPointStruct > stdair::BreakPointList_T

33.244 stdair/bom/BreakPointTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BREAKPOINTTYPES_HPP
00003 #define __STDAIR_BOM_BREAKPOINTTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015   // Forward declarations
00016   struct BreakPointStruct;
00017
00018   // ************* Type definitions **************
00020   typedef boost::shared_ptr<BreakPointStruct> BreakPointPtr_T;
00021
00023   typedef std::list<BreakPointStruct> BreakPointList_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_BREAKPOINTTYPES_HPP
00027
```

33.245 stdair/bom/Bucket.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/Bucket.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.246 stdair/bom/Bucket.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/Bucket.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  Bucket::Bucket()
    : _key (DEFAULT_SEAT_INDEX), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  Bucket::Bucket (const Bucket& iBucket) :
    _key (iBucket._key),
    _parent (NULL),
    _yieldRangeUpperValue (iBucket._yieldRangeUpperValue),
    _availability (iBucket._availability),
    _soldSeats (iBucket._soldSeats) {

  }

  // ////////////////////////////////////////////////////////////////////
  Bucket::Bucket (const Key_T& iKey) : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  Bucket::~Bucket() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string Bucket::toString() const {
    std::ostringstream oStr;
    oStr <<  describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void Bucket::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void Bucket::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void Bucket::serialize (Archive& ioArchive, const unsigned int iFileVersion) {
    ioArchive & _key;
```

```
00066   }
00067
00068 }
00069
```

33.247 stdair/bom/Bucket.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BucketKey.hpp>
#include <stdair/bom/BucketTypes.hpp>
```

Classes

- class stdair::Bucket

  Class representing the actual attributes for an airline booking class.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.248 stdair/bom/Bucket.hpp

```
00001 #ifndef __STDAIR_BOM_BUCKET_HPP
00002 #define __STDAIR_BOM_BUCKET_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BucketKey.hpp>
00014 #include <stdair/bom/BucketTypes.hpp>
00015
00017 namespace boost {
00018   namespace serialization {
00019     class access;
00020   }
00021 }
00022
00023 namespace stdair {
00024
00029   class Bucket : public BomAbstract {
00030     template <typename BOM> friend class FacBom;
00031     template <typename BOM> friend class FacCloneBom;
00032     friend class FacBomManager;
00033     friend class boost::serialization::access;
00034
00035   public:
00036     // ///////////////// Type definitions ////////////////////
00040     typedef BucketKey Key_T;
00041
00042   public:
00043     // /////////// Getters ////////////
00047     const Key_T& getKey() const {
00048       return _key;
00049     }
00050
00054     BomAbstract* const getParent() const {
00055       return _parent;
00056     }
00057
00059     const HolderMap_T& getHolderMap() const {
00060       return _holderMap;
00061     }
00062
00064     const SeatIndex_T& getSeatIndex() const {
00065       return _key.getSeatIndex();
00066     }
00067
00069     const Yield_T& getYieldRangeUpperValue() const {
00070       return _yieldRangeUpperValue;
00071     }
00072
00074     const CabinCapacity_T& getAvailability() const {
00075       return _availability;
00076     }
00077
00079     const NbOfSeats_T& getSoldSeats() const {
00080       return _soldSeats;
00081     }
00082
00083
00084     // /////////// Setters ////////////
```

```
00086     void setYieldRangeUpperValue (const Yield_T& iYield) {
00087       _yieldRangeUpperValue = iYield;
00088     }
00089
00091     void setAvailability (const CabinCapacity_T& iAvl) {
00092       _availability = iAvl;
00093     }
00094
00096     void setSoldSeats (const NbOfSeats_T& iSoldSeats) {
00097       _soldSeats = iSoldSeats;
00098     }
00099
00100
00101   public:
00102     // /////////// Display support methods /////////
00108     void toStream (std::ostream& ioOut) const {
00109       ioOut << toString();
00110     }
00111
00117     void fromStream (std::istream& ioIn) {
00118     }
00119
00123     std::string toString() const;
00124
00128     const std::string describeKey() const {
00129       return _key.toString();
00130     }
00131
00132
00133   public:
00134     // /////////// (Boost) Serialisation support methods /////////
00138     template<class Archive>
00139     void serialize (Archive& ar, const unsigned int iFileVersion);
00140
00141   private:
00146     void serialisationImplementationExport() const;
00147     void serialisationImplementationImport();
00148
00149
00150   protected:
00151     // ////////// Constructors and destructors /////////
00155     Bucket (const Key_T&);
00156
00160     virtual ~Bucket();
00161
00162   private:
00166     Bucket();
00167
00171     Bucket (const Bucket&);
00172
00173
00174   protected:
00175     // //////////////////// Children ////////////////////
00179     Key_T _key;
00180
00184     BomAbstract* _parent;
00185
00189     HolderMap_T _holderMap;
00190
00191
00192   protected:
00193     // //////////////////// Attributes ////////////////////
00197     Yield_T _yieldRangeUpperValue;
00198
00202     CabinCapacity_T _availability;
00203
00207     NbOfSeats_T _soldSeats;
```

```
  };

}
#endif // __STDAIR_BOM_BUCKET_HPP

```

33.249 stdair/bom/BucketKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/bom/BucketKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::BucketKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BucketKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.250 stdair/bom/BucketKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/bom/BucketKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  BucketKey::BucketKey() {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  BucketKey::BucketKey (const SeatIndex_T& iSeatIndex)
    : _seatIndex (iSeatIndex) {
  }

  // ////////////////////////////////////////////////////////////////////
  BucketKey::BucketKey (const BucketKey& iBucketKey)
    : _seatIndex (iBucketKey._seatIndex) {
  }

  // ////////////////////////////////////////////////////////////////////
  BucketKey::~BucketKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void BucketKey::toStream (std::ostream& ioOut) const {
    ioOut << "BucketKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void BucketKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string BucketKey::toString() const {
    std::ostringstream oStr;
    oStr << _seatIndex;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void BucketKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void BucketKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
```

```
00066   template<class Archive>
00067   void BucketKey::serialize (Archive& ioArchive,
00068                              const unsigned int iFileVersion) {
00069     ioArchive & _seatIndex;
00070   }
00071
00072   // ////////////////////////////////////////////////////////////////////
00073   // Explicit template instantiation
00074   namespace ba = boost::archive;
00075   template void BucketKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00076                                                          unsigned int);
00077   template void BucketKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00078                                                          unsigned int);
00079   // ////////////////////////////////////////////////////////////////////
00080
00081 }
```

33.251 stdair/bom/BucketKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::BucketKey

  Key of booking-class.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.252 stdair/bom/BucketKey.hpp

```
00001 #ifndef __STDAIR_BOM_BUCKETKEY_HPP
00002 #define __STDAIR_BOM_BUCKETKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct BucketKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     BucketKey();
00035
00036   public:
00040     BucketKey (const SeatIndex_T&);
00044     BucketKey (const BucketKey&);
00048     ~BucketKey();
00049
00050
00051   public:
00052     // /////////// Getters //////////
00054     const SeatIndex_T& getSeatIndex() const {
00055       return _seatIndex;
00056     }
00057
00058
00059   public:
00060     // /////////// Display support methods /////////
00066     void toStream (std::ostream& ioOut) const;
00067
00073     void fromStream (std::istream& ioIn);
00074
00084     const std::string toString() const;
00085
00086
00087   public:
00088     // /////////// (Boost) Serialisation support methods /////////
00092     template<class Archive>
00093     void serialize (Archive& ar, const unsigned int iFileVersion);
00094
00095   private:
00100     void serialisationImplementationExport() const;
00101     void serialisationImplementationImport();
00102
00103
00104   private:
00105     // ////////////////// Attributes ///////////////
00109     SeatIndex_T _seatIndex;
00110   };
00111
```

```
}
#endif // __STDAIR_BOM_BUCKETKEY_HPP
```

33.253 stdair/bom/BucketTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< Bucket * > stdair::BucketList_T
- typedef std::map< const MapKey_T, Bucket * > stdair::BucketMap_T

33.254 stdair/bom/BucketTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_BUCKETTYPES_HPP
00003 #define __STDAIR_BOM_BUCKETTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations
00017   class Bucket;
00018
00020   typedef std::list<Bucket*> BucketList_T;
00021
00023   typedef std::map<const MapKey_T, Bucket*> BucketMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_BUCKETTYPES_HPP
00027
```

33.255 stdair/bom/CancellationStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.256 stdair/bom/CancellationStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/BookingClass.hpp>

namespace stdair {
  // ////////////////////////////////////////////////////////////////////
  CancellationStruct::CancellationStruct (const SegmentPath_T& iSegPath,
                                          const ClassList_String_T& iList,
                                          const PartySize_T& iSize,
                                          const DateTime_T& iDateTime)
    : _segmentPath (iSegPath), _classList (iList), _partySize (iSize),
      _datetime (iDateTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  CancellationStruct::CancellationStruct (const SegmentPath_T& iSegPath,
                                          const BookingClassIDList_T& iIDList,
                                          const PartySize_T& iSize,
                                          const DateTime_T& iDateTime)
    : _segmentPath (iSegPath), _classIDList (iIDList), _partySize (iSize),
      _datetime (iDateTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  CancellationStruct::~CancellationStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  void CancellationStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void CancellationStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string CancellationStruct::describe() const {
    std::ostringstream oStr;

    oStr << "Segment path: ";
    unsigned short idx = 0;
    for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
         lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
      if (idx != 0) {
        oStr << "-";
      }
      const std::string& lSegmentKey = *lItSegmentPath;
      oStr << lSegmentKey;
    }
    if (_classList == "") {
      oStr << ";";
      BookingClassIDList_T::const_iterator lItBookingClassIDList =
        _classIDList.begin();
      idx = 0;
      for (; lItBookingClassIDList != _classIDList.end();
             ++lItBookingClassIDList, ++idx) {
        if (idx != 0) {
```

```
          oStr << "-";
        }
        const BookingClassID_T& lBookingClassID = *lItBookingClassIDList;
        const BookingClass& lBookingClass = lBookingClassID.getObject();
        const ClassCode_T& lClassCode = lBookingClass.getClassCode();
        oStr << lClassCode;
      }
      oStr << ";" << _partySize << ";" << _datetime;
    } else {
      oStr << ";" << _classList << ";" << _partySize << ";" << _datetime;
    }
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string CancellationStruct::display() const {
    std::ostringstream oStr;

    // List of segment keys (one per segment)
    unsigned short idx = 0;
    for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
         itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
      if (idx != 0) {
        oStr << " ; ";
      }
      const std::string& lSegmentKey = *itSegPath;
      oStr << "[" << idx << "] " << lSegmentKey;
    }
    if (_classList == "") {
      oStr << ";";
      BookingClassIDList_T::const_iterator lItBookingClassIDList =
        _classIDList.begin();
      idx = 0;
      for (; lItBookingClassIDList != _classIDList.end();
           ++lItBookingClassIDList, ++idx) {
        if (idx != 0) {
          oStr << "-";
        }
        const BookingClassID_T& lBookingClassID = *lItBookingClassIDList;
        const BookingClass& lBookingClass = lBookingClassID.getObject();
        const ClassCode_T& lClassCode = lBookingClass.getClassCode();
        oStr << lClassCode;
      }
      oStr << ";" << _partySize << ";" << _datetime;
    } else {
      oStr << ";" << _classList << ";" << _partySize << ";" << _datetime;
    }
    return oStr.str();
  }
}
```

33.257 stdair/bom/CancellationStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/BomIDTypes.hpp>
```

Classes

- struct stdair::CancellationStruct

  Structure holding the elements of a travel solution.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.258 stdair/bom/CancellationStruct.hpp

```
00001 #ifndef __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00002 #define __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 #include <vector>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/BookingClassTypes.hpp>
00015 #include <stdair/bom/TravelSolutionTypes.hpp>
00016 #include <stdair/bom/BomIDTypes.hpp>
00017
00018 namespace stdair {
00019
00023   struct CancellationStruct : public StructAbstract {
00024   public:
00025     // /////////// Getters ///////////////
00027     const SegmentPath_T& getSegmentPath() const {
00028       return _segmentPath;
00029     }
00030
00032     const ClassList_String_T& getClassList() const {
00033       return _classList;
00034     }
00035
00037     const BookingClassIDList_T& getClassIDList() const {
00038       return _classIDList;
00039     }
00040
00042     const PartySize_T& getPartySize() const {
00043       return _partySize;
00044     }
00045
00047     const DateTime_T& getCancellationDateTime() const {
00048       return _datetime;
00049     }
00050
00051   public:
00052     // /////////// Display support method /////////////
00058     void toStream (std::ostream& ioOut) const;
00059
00064     void fromStream (std::istream& ioIn);
00065
00069     const std::string describe() const;
00070
00074     const std::string display() const;
00075
00076
00077   public:
00078     // ////////////// Constructors & Destructor ///////////////
00082     CancellationStruct (const SegmentPath_T&, const ClassList_String_T&,
00083                         const PartySize_T&, const DateTime_T&);
00084
00088     CancellationStruct (const SegmentPath_T&, const BookingClassIDList_T&,
00089                         const PartySize_T&, const DateTime_T&);
00090
00094     ~CancellationStruct();
00095
00096
00097   private:
```

```
    // ////////////////////// Attributes ///////////////////////
    SegmentPath_T _segmentPath;

    ClassList_String_T _classList;

    BookingClassIDList_T _classIDList;

    PartySize_T _partySize;

    DateTime_T _datetime;
  };

}
#endif // __STDAIR_BOM_CANCELLATIONSTRUCT_HPP
```

33.259 stdair/bom/CancellationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< CancellationStruct > stdair::CancellationPtr_T

33.260 stdair/bom/CancellationTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_CANCELLATIONTYPES_HPP
#define __STDAIR_BOM_CANCELLATIONTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// Boost
#include <boost/shared_ptr.hpp>

namespace stdair {

  // Forward declarations
  struct CancellationStruct;

  // ///////////// Type definitions //////////////
  typedef boost::shared_ptr<CancellationStruct> CancellationPtr_T;

}
#endif // __STDAIR_BOM_CANCELLATIONTYPES_HPP

```

33.261 stdair/bom/ConfigHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.262 stdair/bom/ConfigHolderStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
#if BOOST_VERSION >= 104100
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/json_parser.hpp>
#include <boost/foreach.hpp>
#endif // BOOST_VERSION >= 104100
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/ForecastingMethod.hpp>
#include <stdair/basic/UnconstrainingMethod.hpp>
#include <stdair/basic/PartnershipTechnique.hpp>
#include <stdair/basic/PreOptimisationMethod.hpp>
#include <stdair/basic/OptimisationMethod.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct::ConfigHolderStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct::
  ConfigHolderStruct (const ConfigHolderStruct& iConfigHolderStruct)
    : _pt (iConfigHolderStruct._pt) {
  }

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct::~ConfigHolderStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  void ConfigHolderStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void ConfigHolderStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string ConfigHolderStruct::describe() const {
    std::ostringstream oStr;
    oStr << "Configuration Display:" << std::endl;

    // Look for the start and end date values.
    stdair::Date_T lStartDate;
    const bool hasStartDateBeenRetrieved =
      exportValue<Date_T> (lStartDate, "date.start");
    if (hasStartDateBeenRetrieved == true) {
      oStr << "  Start date: " << lStartDate << std::endl;
    }
    stdair::Date_T lEndDate;
    const bool hasEndDateBeenRetrieved =
      exportValue<Date_T> (lEndDate, "date.end");
    if (hasEndDateBeenRetrieved == true) {
      oStr << "  End date: " << lEndDate << std::endl;
```

```
    }

    // Look for the random seed value.
    RandomSeed_T lRandomSeed;
    const bool hasSeedBeenRetrieved =
      exportValue<RandomSeed_T> (lRandomSeed, "random.seed");
    if (hasSeedBeenRetrieved == true) {
      oStr << "  Random Seed: " << lRandomSeed << std::endl;
    }

    // Look for the demand generation method.
    char lChar;
    const bool hasDemandGenMethodBeenRetrieved =
      exportValue<char> (lChar, "demand generation.method");
    if (hasDemandGenMethodBeenRetrieved == true) {
      oStr << "  Demand Generation method: " << lChar << std::endl;
    }

    // Look for the number of runs value.
    Count_T lTotalNumberOfRuns;
    const bool hasNumberOfRunsBeenRetrieved =
      exportValue<Count_T> (lTotalNumberOfRuns, "runs.number");
    if (hasNumberOfRunsBeenRetrieved == true) {
      oStr << "  Number Of Runs: " << lTotalNumberOfRuns << std::endl;
    }

    // Look for the input files.
    stdair::Filename_T lFilename ("");
    const bool hasScheduleFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.schedule");
    if (hasScheduleFileBeenRetrieved == true) {
      oStr << "  Schedule input file: " << lFilename << std::endl;
    }
    const bool hasODFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.ond");
    if (hasODFileBeenRetrieved == true) {
      oStr << "  OnD input file: " << lFilename << std::endl;
    }
    const bool hasFrat5FileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.frat5");
    if (hasFrat5FileBeenRetrieved == true) {
      oStr << "  Frat5 input file: " << lFilename << std::endl;
    }
    const bool hasFFdisutilityFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.ffdisutility");
    if (hasFFdisutilityFileBeenRetrieved == true) {
      oStr << "  FFdisutility input file: " << lFilename << std::endl;
    }
    const bool hasYieldFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.yield");
    if (hasYieldFileBeenRetrieved == true) {
      oStr << "  Yield input file: " << lFilename << std::endl;
    }
    const bool hasFareFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.fare");
    if (hasFareFileBeenRetrieved == true) {
      oStr << "  Fare input file: " << lFilename << std::endl;
    }
    const bool hasDemandFileBeenRetrieved =
      exportValue<stdair::Filename_T> (lFilename, "input.demand");
    if  (hasDemandFileBeenRetrieved == true) {
      oStr << "  Demand input file: " << lFilename << std::endl;
    }

    return oStr.str();
  }

```

```
  // ////////////////////////////////////////////////////////////////////
  const std::string ConfigHolderStruct::jsonExport() const {
    std::ostringstream oStr;
#if BOOST_VERSION >= 104100
    // Write the property tree into the JSON stream.
    write_json (oStr, _pt);
#endif // BOOST_VERSION >= 104100
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void ConfigHolderStruct::add (const bpt::ptree& iConfigPropertyTree) {
    // Call the dedicated recursive method with an empty path in order to merge
    // the config property tree with the given new one.
    std::string lEmptyPath ("");
    add (iConfigPropertyTree, lEmptyPath);
  }

  // ////////////////////////////////////////////////////////////////////
  void ConfigHolderStruct::add (const bpt::ptree& iConfigPropertyTree,
                                const std::string& iPath) {

    // Are there any more children to browse?
    bool isThereAnyChild = false;

#if BOOST_VERSION >= 104100

    // Browse the children nodes
    BOOST_FOREACH(bpt::ptree::value_type itChild, iConfigPropertyTree) {

      isThereAnyChild = true;

      // Build the current path
      std::ostringstream lCurrentPathStr;
      const bool isPathEmptyForNow = iPath.empty();
      if (isPathEmptyForNow == false) {
        lCurrentPathStr << iPath << ".";
      }
      // Add the current node name
      lCurrentPathStr << itChild.first.data();
      const std::string lCurrentPath (lCurrentPathStr.str());

      // Get the child tree
      const bpt::ptree& lChildTree = itChild.second;
      add(lChildTree, lCurrentPath);
    }

    // If there is no child for this node, create the specified path and add
    // the correponding value
    if (isThereAnyChild == false) {
      std::string lValue (iConfigPropertyTree.data());
      const bool hasInsertionBeenSuccessful = addValue (lValue, iPath);
      assert (hasInsertionBeenSuccessful == true);
    }
#endif // BOOST_VERSION >= 104100
  }

  // ////////////////////////////////////////////////////////////////////
  bool ConfigHolderStruct::addValue (const std::string& iValue,
                                     const std::string& iPath) {
    bool hasInsertionBeenSuccessful = true;
    // Create the given specified path and add the corresponding given value,
    // or replace the value if the path already exists.
#if BOOST_VERSION >= 104100

    try {
      std::size_t found;
```

```
      const std::string lPrefix ("config");
      std::string lFinalPath;
      found = iPath.find(lPrefix);
      if (found == std::string::npos) {
        lFinalPath += lPrefix;
        lFinalPath += ".";
      }
      lFinalPath += iPath;
      if (lFinalPath != lPrefix) {
        _pt.put (lFinalPath, iValue);
      }
    } catch (bpt::ptree_bad_data& bptException) {
      hasInsertionBeenSuccessful = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasInsertionBeenSuccessful;
  }

  // ////////////////////////////////////////////////////////////////////
  void ConfigHolderStruct::updateAirlineFeatures (BomRoot& iBomRoot) {

    AirlineCode_T lAirlineCode ("");

    // Browse the children nodes
    BOOST_FOREACH(bpt::ptree::value_type itChild, _pt) {
      std::ostringstream lPathStr;
      lPathStr << itChild.first.data() << ".airline_code";
      const bool hasAirlineCodeBeenRetrieved =
        exportValue<AirlineCode_T> (lAirlineCode , lPathStr.str());
      if (hasAirlineCodeBeenRetrieved == true) {
        AirlineFeature* lAirlineFeature_ptr =
          BomRetriever::retrieveAirlineFeatureFromKey (iBomRoot, lAirlineCode);
        if (lAirlineFeature_ptr != NULL) {

          try {

            std::ostringstream lPathStr;
            char lChar;

            // Try to extract the forecasting method from the config tree
            lPathStr << itChild.first.data() << ".forecasting_method";
            const bool hasForecastingMethodBeenRetrieved =
              exportValue<char> (lChar, lPathStr.str());
            if (hasForecastingMethodBeenRetrieved == true) {
              const ForecastingMethod lForecastingMethod (lChar);
              lAirlineFeature_ptr->setForecastingMethod(lForecastingMethod);
            }

            // Try to extract the unconstraining method from the config tree
            lPathStr.str("");
            lPathStr << itChild.first.data() << ".unconstraining_method";
            const bool hasUnconstrainingMethodBeenRetrieved =
              exportValue<char> (lChar, lPathStr.str());
            if (hasUnconstrainingMethodBeenRetrieved == true) {
              const UnconstrainingMethod lUnconstrainingMethod (lChar);
              lAirlineFeature_ptr->setUnconstrainingMethod(lUnconstrainingMethod);
            }

            // Try to extract the partnership technique from the config tree
            lPathStr.str("");
            lPathStr << itChild.first.data() << ".partnership_technique";
            const bool hasPartnershipTechniqueBeenRetrieved =
              exportValue<char> (lChar, lPathStr.str());
            if (hasPartnershipTechniqueBeenRetrieved == true) {
              const PartnershipTechnique lPartnershipTechnique (lChar);
```

```
00266                 lAirlineFeature_ptr->setPartnershipTechnique(lPartnershipTechnique)
      ;
00267               }
00268
00269               // Try to extract the pre optimisation method from the config tree
00270               lPathStr.str("");
00271               lPathStr << itChild.first.data() << ".pre_optimisation_method";
00272               const bool hasPreOptMethodBeenRetrieved =
00273                 exportValue<char> (lChar, lPathStr.str());
00274               if (hasPreOptMethodBeenRetrieved == true) {
00275                 const PreOptimisationMethod lPreOptimisationMethod (lChar);
00276                 lAirlineFeature_ptr->setPreOptimisationMethod(lPreOptimisationMetho
      d);
00277               }
00278
00279               // Try to extract the optimisation method from the config tree
00280               lPathStr.str("");
00281               lPathStr << itChild.first.data() << ".optimisation_method";
00282               const bool hasOptMethodBeenRetrieved =
00283                 exportValue<char> (lChar, lPathStr.str());
00284               if (hasOptMethodBeenRetrieved == true) {
00285                 const OptimisationMethod lOptimisationMethod (lChar);
00286                 lAirlineFeature_ptr->setOptimisationMethod(lOptimisationMethod);
00287               }
00288
00289             } catch (CodeConversionException& lCodeConversionException) {
00290               std::ostringstream oMessage;
00291               oMessage << "Wrong input features for the airline '"
00292                        << lAirlineCode << "' in the input configuration file: "
00293                        << lCodeConversionException.what();
00294               STDAIR_LOG_ERROR (oMessage.str());
00295               throw CodeConversionException (oMessage.str());
00296             }
00297           }
00298         }
00299       }
00300     }
00301 }
```

33.263 stdair/bom/ConfigHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/ConfigHolderTypes.hpp>
```

Classes

- struct stdair::ConfigHolderStruct

Namespaces

- namespace bpt
- namespace stdair

  Handle on the StdAir library context.

33.264 stdair/bom/ConfigHolderStruct.hpp

```
00001 #ifndef __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP
00002 #define __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // Boost
00011 #include <boost/static_assert.hpp>
00012 #include <boost/type_traits/is_same.hpp>
00013 #if BOOST_VERSION >= 104100
00014 // Boost Property Tree
00015 #include <boost/property_tree/ptree.hpp>
00016 #endif // BOOST_VERSION >= 104100
00017 // StdAir
00018 #include <stdair/stdair_file.hpp>
00019 #include <stdair/stdair_maths_types.hpp>
00020 #include <stdair/stdair_date_time_types.hpp>
00021 #include <stdair/basic/StructAbstract.hpp>
00022 #include <stdair/bom/ConfigHolderTypes.hpp>
00023
00024 #if BOOST_VERSION >= 104100
00025 namespace bpt = boost::property_tree;
00026 #else // BOOST_VERSION >= 104100
00027 namespace bpt {
00028   typedef char ptree;
00029 }
00030 #endif // BOOST_VERSION >= 104100
00031
00032 namespace stdair {
00033
00035   class BomRoot;
00036
00040   struct ConfigHolderStruct : public StructAbstract {
00041   public:
00042     // /////////////// Getters /////////////////
00043
00044     // ////////////// Business Methods //////////
00051     void add (const bpt::ptree&);
00052
00061     bool addValue (const std::string& iValue,
00062                    const std::string& iPath);
00063
00072     template <typename ValueType>
00073     bool exportValue (ValueType& ioValue, const std::string& iPath) const;
00074
00081     void updateAirlineFeatures (BomRoot&);
00082
00083   private:
00087     void add (const bpt::ptree&,
00088               const std::string&);
00089
00090   public:
00091     // /////////// Display support method /////////////
00096     void toStream (std::ostream& ioOut) const;
00097
00102     void fromStream (std::istream& ioIn);
00103
00107     const std::string describe() const;
00108
00112     const std::string jsonExport() const;
00113
00114
```

```
    // /////////////// Constructors and Destructors /////////////////
  public:
    ConfigHolderStruct ();

    ConfigHolderStruct (const ConfigHolderStruct&);

  public:
    ~ConfigHolderStruct();


  private:
    // /////////////// Attributes /////////////////
    bpt::ptree _pt;
  };

  // ////////////////////////////////////////////////////////////////////
  template <typename ValueType>
  bool ConfigHolderStruct::exportValue (ValueType& ioValue,
                                        const std::string& iPath) const {

    bool hasValueBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100
    try {
      // Get the value.
      // If the path key is not found, an exception is thrown.
      const std::string lPrefix ("config.");
      const std::string lFinalPath = lPrefix + iPath;
      ioValue = _pt.get<ValueType> (lFinalPath);

    } catch (bpt::ptree_error& bptException) {
      hasValueBeenSuccessfullyRetrieved = false;
    }
#endif // BOOST_VERSION >= 104100

    return hasValueBeenSuccessfullyRetrieved;

  }

  // ////////////////////////////////////////////////////////////////////
  //
  // Specialization of the template method exportValue above for the type
  // Date_T.
  //
  // ////////////////////////////////////////////////////////////////////

  template<>
  inline bool ConfigHolderStruct::exportValue<Date_T>
  (Date_T& ioValue,
   const std::string& iPath) const {

    bool hasValueBeenSuccessfullyRetrieved = true;

#if BOOST_VERSION >= 104100

    try {

      // Get the string date value.
      // If the path key is not found, an exception is thrown.
      const std::string lPrefix ("config.");
      const std::string lFinalPath = lPrefix + iPath;
      const std::string& lDateStr =
        _pt.get<std::string> (lFinalPath);

      // Convert the string into a Date_T.
      ioValue =
        boost::gregorian::from_simple_string (lDateStr);
```

```
00194
00195     } catch (bpt::ptree_error& bptException) {
00196       hasValueBeenSuccessfullyRetrieved = false;
00197     }
00198 #endif // BOOST_VERSION >= 104100
00199
00200     return hasValueBeenSuccessfullyRetrieved;
00201
00202
00203   }
00204
00205 }
00206
00207 #endif // __STDAIR_BOM_CONFIGHOLDERSTRUCT_HPP
```

33.265 stdair/bom/ConfigHolderTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< ConfigHolderStruct > stdair::ConfigHolderPtr_T

33.266 stdair/bom/ConfigHolderTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00003 #define __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015   // Forward declarations
00016   struct ConfigHolderStruct;
00017
00018   // ////////////// Type definitions //////////////
00020   typedef boost::shared_ptr<ConfigHolderStruct> ConfigHolderPtr_T;
00021
00022 }
00023 #endif // __STDAIR_BOM_CONFIGHOLDERTYPES_HPP
00024
```

33.267 stdair/bom/DatePeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/DatePeriod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.268 stdair/bom/DatePeriod.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/DatePeriod.hpp>
00011
00012 namespace stdair {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   DatePeriod::DatePeriod()
00016     : _key (BOOST_DEFAULT_DATE_PERIOD),
00017       _parent (NULL)  {
00018     // That constructor is used by the serialisation process
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   DatePeriod::DatePeriod (const DatePeriod& iDatePeriod)
00023     : _key (iDatePeriod.getKey()), _parent (NULL)  {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   DatePeriod::DatePeriod (const Key_T& iKey)
00028     : _key (iKey), _parent (NULL)  {
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   DatePeriod::~DatePeriod () {
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   std::string DatePeriod::toString() const {
00037     std::ostringstream oStr;
00038     oStr << describeKey();
00039     return oStr.str();
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   bool DatePeriod::
00044   isDepartureDateValid (const Date_T& iFlightDate) const {
00045
00046     // Check if the departure date is within the date range.
00047     const DatePeriod_T& lPeriod = getDatePeriod ();
00048     if (lPeriod.contains (iFlightDate) == false) {
00049       return false;
00050     }
00051
00052     return true;
00053   }
00054
00055 }
00056
```

33.269 stdair/bom/DatePeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
#include <stdair/bom/DatePeriodTypes.hpp>
```

Classes

- class stdair::DatePeriod

  Class representing the actual attributes for a fare date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.270 stdair/bom/DatePeriod.hpp

```
00001 #ifndef __STDAIR_BOM_DATEPERIOD_HPP
00002 #define __STDAIR_BOM_DATEPERIOD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/DatePeriodKey.hpp>
00010 #include <stdair/bom/DatePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018   class DatePeriod : public BomAbstract {
00019     template <typename BOM> friend class FacBom;
00020     template <typename BOM> friend class FacCloneBom;
00021     friend class FacBomManager;
00022
00023   public:
00024     // ////////////// Type definitions //////////////
00028     typedef DatePeriodKey Key_T;
00029
00030   public:
00031     // /////////// Display support methods /////////
00037     void toStream (std::ostream& ioOut) const {
00038       ioOut << toString();
00039     }
00040
00046     void fromStream (std::istream& ioIn) {
00047     }
00048
00052     std::string toString() const;
00053
00057     const std::string describeKey() const {
00058       return _key.toString();
00059     }
00060
00061   public:
00062     // /////////// Getters ////////////
00066     const Key_T& getKey() const {
00067       return _key;
00068     }
00069
00073     BomAbstract* const getParent() const {
00074       return _parent;
00075     }
00076
00080     const  HolderMap_T& getHolderMap() const {
00081       return _holderMap;
00082     }
00083
00087     const DatePeriod_T& getDatePeriod() const {
00088       return _key.getDatePeriod();
00089     }
00090
00091
00092   public:
00093     // ////////////// Business methods ///////////////
00098     bool isDepartureDateValid (const Date_T&) const;
00099
00100   protected:
00101     // /////////// Constructors and destructors /////////
00105     DatePeriod (const Key_T&);
00109     virtual ~DatePeriod ();
```

```
00110
00111   private:
00115     DatePeriod ();
00119     DatePeriod (const DatePeriod&);
00120
00121   protected:
00122     // ############# Attributes #############
00126     Key_T _key;
00127
00131     BomAbstract* _parent;
00132
00136     HolderMap_T _holderMap;
00137
00138   };
00139
00140 }
00141 #endif // __STDAIR_BOM_DATEPERIOD_HPP
00142
```

33.271 stdair/bom/DatePeriodKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DatePeriodKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.272 stdair/bom/DatePeriodKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// Boost Date-Time
#include <boost/date_time/gregorian/formatters.hpp>
// STDAIR
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DatePeriodKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  DatePeriodKey::DatePeriodKey()
    : _datePeriod (BOOST_DEFAULT_DATE_PERIOD) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  DatePeriodKey::DatePeriodKey (const stdair::DatePeriod_T& iDatePeriod)
    : _datePeriod (iDatePeriod) {
  }

  // ////////////////////////////////////////////////////////////////////
  DatePeriodKey::DatePeriodKey (const DatePeriodKey& iKey)
    : _datePeriod (iKey._datePeriod) {
  }

  // ////////////////////////////////////////////////////////////////////
  DatePeriodKey::~DatePeriodKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void DatePeriodKey::toStream (std::ostream& ioOut) const {
    ioOut << "DatePeriodKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void DatePeriodKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string DatePeriodKey::toString() const {
    std::ostringstream oStr;
    const stdair::Date_T lStart = _datePeriod.begin();
    const stdair::Date_T lEnd = _datePeriod.end();
    oStr << "[" << boost::gregorian::to_simple_string(lStart)
         << "/" << boost::gregorian::to_simple_string(lEnd)
         << "]";
    return oStr.str();
  }

}
```

33.273 stdair/bom/DatePeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct stdair::DatePeriodKey

  Key of date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.274 stdair/bom/DatePeriodKey.hpp

```
00001 #ifndef __SIMFQT_BOM_DATEPERIODKEY_HPP
00002 #define __SIMFQT_BOM_DATEPERIODKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010
00011 namespace stdair  {
00014   struct DatePeriodKey : public KeyAbstract {
00015
00016   public:
00017     // /////////// Construction ///////////
00019     DatePeriodKey (const DatePeriod_T&);
00021     DatePeriodKey (const DatePeriodKey&);
00023     ~DatePeriodKey ();
00024
00025   private:
00027     DatePeriodKey();
00028
00029   public:
00030     // /////////// Getters //////////
00032     const DatePeriod_T& getDatePeriod() const {
00033       return _datePeriod;
00034     }
00035
00036   public:
00037
00038     // /////////// Display support methods /////////
00044     void toStream (std::ostream& ioOut) const;
00045
00051     void fromStream (std::istream& ioIn);
00052
00058     const std::string toString() const;
00059
00060   private:
00061     // ///////////////// Attributes ///////////////////
00065     DatePeriod_T _datePeriod;
00066
00067   };
00068
00069 }
00070 #endif // __SIMFQT_BOM_DATEPERIODKEY_HPP
```

33.275 stdair/bom/DatePeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< DatePeriod * > stdair::DatePeriodList_T
- typedef std::map< const MapKey_T, DatePeriod * > stdair::DatePeriodMap_T
- typedef std::pair< MapKey_T, DatePeriod * > stdair::DatePeriodWithKey_T
- typedef std::list< DatePeriodWithKey_T > stdair::DatePeriodDetailedList_T

33.276 stdair/bom/DatePeriodTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_DATEPERIODTYPES_HPP
#define __STDAIR_BOM_DATEPERIODTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class DatePeriod;

  typedef std::list<DatePeriod*> DatePeriodList_T;

  typedef std::map<const MapKey_T, DatePeriod*> DatePeriodMap_T;

  typedef std::pair<MapKey_T, DatePeriod*> DatePeriodWithKey_T;
  typedef std::list<DatePeriodWithKey_T> DatePeriodDetailedList_T;
}
#endif // __STDAIR_BOM_DATEPERIODTYPES_HPP

```

33.277 stdair/bom/DoWStruct.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/DoWStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.278 stdair/bom/DoWStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // STDAIR
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/bom/DoWStruct.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   DoWStruct::DoWStruct () {
00015     for (unsigned short i = 0; i < 7; ++i) {
00016       _dowList.push_back (false);
00017     }
00018   }
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   DoWStruct::DoWStruct (const std::string& iDowString) {
00022     const unsigned short lDowStringSize = iDowString.size();
00023     assert (lDowStringSize == 7);
00024
00025     _dowList.reserve (lDowStringSize);
00026     for (std::string::const_iterator itChar = iDowString.begin();
00027          itChar != iDowString.end(); ++itChar) {
00028       const bool isDoWSet = (*itChar == '1')?true:false;
00029       _dowList.push_back (isDoWSet);
00030     }
00031   }
00032
00033   // ////////////////////////////////////////////////////////////////////
00034   DoWStruct::DoWStruct (const DoWStruct& iDowStruct) :
00035     _dowList (iDowStruct._dowList) {
00036
00037   }
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   const std::string DoWStruct::describeShort() const {
00041     std::ostringstream ostr;
00042     short i = 0;
00043     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00044          itDoW != _dowList.end(); ++itDoW, ++i) {
00045       const char lDoW = (*itDoW == true)?'1':'0';
00046       ostr << lDoW;
00047     }
00048     return ostr.str();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   const std::string DoWStruct::describe() const {
00053     std::ostringstream ostr;
00054     short i = 0;
00055     for (BooleanList_T::const_iterator itDoW = _dowList.begin();
00056          itDoW != _dowList.end(); ++itDoW, ++i) {
00057       const bool lDoW = *itDoW;
00058       if (lDoW == true) {
00059         ostr << DOW_STR[i] << ".";
00060       }
00061     }
00062     return ostr.str();
00063   }
00064
00065   // ////////////////////////////////////////////////////////////////////
```

```
00066   bool DoWStruct::getDayOfWeek (const unsigned short i) const {
00067     return _dowList.at (i);
00068   }
00069
00070   // ////////////////////////////////////////////////////////////////////
00071   bool DoWStruct::getStandardDayOfWeek (const unsigned short i) const {
00072     unsigned short iStd = i;
00073     if (iStd == 0) {
00074       iStd = 6;
00075     } else {
00076       --iStd;
00077     }
00078     return _dowList.at (iStd);
00079   }
00080
00081   // ////////////////////////////////////////////////////////////////////
00082   void DoWStruct::setDayOfWeek (const unsigned short i, const bool iBool) {
00083     assert (i < 7);
00084     _dowList.at (i) = iBool;
00085   }
00086
00087   // ////////////////////////////////////////////////////////////////////
00088   DoWStruct DoWStruct::shift (const long& iNbOfDays) const {
00089     DoWStruct oDoW (DEFAULT_DOW_STRING);
00090
00091     for (short i = 0; i < 7; ++i) {
00092       const bool lDoWBool = _dowList.at (i);
00093       short lIndex = (i + iNbOfDays) % 7;
00094       if (lIndex < 0) {
00095         lIndex += 7;
00096       }
00097       oDoW.setDayOfWeek (lIndex, lDoWBool);
00098     }
00099
00100     return oDoW;
00101   }
00102
00103   // ////////////////////////////////////////////////////////////////////
00104   DoWStruct DoWStruct::intersection (const DoWStruct& iDoW) const {
00105     DoWStruct oDoW (DEFAULT_DOW_STRING);
00106     for (unsigned short i = 0; i < 7; ++i) {
00107       if (getDayOfWeek(i) && iDoW.getDayOfWeek(i)) {
00108         oDoW.setDayOfWeek (i, true);
00109       } else {
00110         oDoW.setDayOfWeek (i, false);
00111       }
00112     }
00113     return oDoW;
00114   }
00115
00116   // ////////////////////////////////////////////////////////////////////
00117   const bool DoWStruct::isValid () const {
00118     for (unsigned short i = 0; i < 7; ++i) {
00119       if (getDayOfWeek(i)) {
00120         return true;
00121       }
00122     }
00123     return false;
00124   }
00125
00126 }
```

33.279 stdair/bom/DoWStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::DoWStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.280 stdair/bom/DoWStruct.hpp

```
00001 #ifndef __STDAIR_BOM_DOWSTRUCT_HPP
00002 #define __STDAIR_BOM_DOWSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 // STDAIR
00011 #include <stdair/basic/StructAbstract.hpp>
00012
00013 namespace stdair {
00014
00018   struct DoWStruct : public StructAbstract {
00019   public:
00021     typedef std::vector<bool> BooleanList_T;
00022
00023   public:
00024     // ///////////////// Getters ///////////////
00026     bool getDayOfWeek (const unsigned short i) const;
00027
00029     bool getStandardDayOfWeek (const unsigned short i) const;
00030
00031   public:
00032     // ///////////////// Setters ///////////////
00034     void setDayOfWeek (const unsigned short, const bool);
00035
00036   public:
00037     // ///////////////// Display methods ///////////////
00039     const std::string describe() const;
00040
00042     const std::string describeShort() const;
00043
00044   public:
00045     // ////////////////// Business Methods ////////////////
00047     DoWStruct shift (const long&) const;
00048
00050     DoWStruct intersection (const DoWStruct&) const;
00051
00053     const bool isValid () const;
00054
00055   public:
00058     DoWStruct (const std::string& iDowString);
00060     DoWStruct ();
00061     DoWStruct (const DoWStruct&);
00063     ~DoWStruct () { }
00064
00065   private:
00067     BooleanList_T _dowList;
00068   };
00069
00070 }
00071 #endif // __STDAIR_BOM_DOWSTRUCT_HPP
```

33.281 stdair/bom/EventStruct.cpp File Reference

```
#include <cassert>
#include <boost/shared_ptr.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/bom/EventStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.282 stdair/bom/EventStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// Boost
#if BOOST_VERSION >= 103900
#include <boost/make_shared.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/shared_ptr.hpp>
#endif // BOOST_VERSION >= 103900
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Event.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
#include <stdair/bom/SnapshotStruct.hpp>
#include <stdair/bom/CancellationStruct.hpp>
#include <stdair/bom/RMEventStruct.hpp>
#include <stdair/bom/BreakPointStruct.hpp>
#include <stdair/bom/EventStruct.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  EventStruct::EventStruct()
    : _eventType (EventType::BKG_REQ), _eventTimeStamp (0) {
  }

  // ////////////////////////////////////////////////////////////////////
  EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
                            BookingRequestPtr_T ioRequestPtr)
    : _eventType (iEventType) {

    //
    assert (ioRequestPtr != NULL);
#if BOOST_VERSION >= 103900
    _bookingRequest = boost::make_shared<BookingRequestStruct> (*ioRequestPtr);
#else  // BOOST_VERSION >= 103900
    _bookingRequest = ioRequestPtr;
#endif // BOOST_VERSION >= 103900
    assert (_bookingRequest != NULL);

    const Duration_T lDuration =
      _bookingRequest->getRequestDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
    _eventTimeStamp = lDuration.total_milliseconds();
  }

  // ////////////////////////////////////////////////////////////////////
  EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
                            CancellationPtr_T ioCancellationPtr)
    : _eventType (iEventType) {

    //
    assert (ioCancellationPtr != NULL);
#if BOOST_VERSION >= 103900
    _cancellation = boost::make_shared<CancellationStruct> (*ioCancellationPtr);
#else  // BOOST_VERSION >= 103900
    _cancellation = ioCancellationPtr;
#endif // BOOST_VERSION >= 103900
    assert (_cancellation != NULL);

    const Duration_T lDuration =
      _cancellation->getCancellationDateTime() - DEFAULT_EVENT_OLDEST_DATETIME;
    _eventTimeStamp = lDuration.total_milliseconds();
```

```
00076   }
00077
00078   // ////////////////////////////////////////////////////////////////////
00079   EventStruct::
00080   EventStruct (const EventType::EN_EventType& iEventType,
00081                const DateTime_T& iDCPDate,
00082                OptimisationNotificationPtr_T ioOptimisationNotificationPtr)
00083     : _eventType (iEventType) {
00084
00085     //
00086     assert (ioOptimisationNotificationPtr != NULL);
00087 #if BOOST_VERSION >= 103900
00088     _optimisationNotification =
00089       boost::make_shared<OptimisationNotificationStruct> (*ioOptimisationNotifica
      tionPtr);
00090 #else  // BOOST_VERSION >= 103900
00091     _optimisationNotification = ioOptimisationNotificationPtr;
00092 #endif // BOOST_VERSION >= 103900
00093     assert (_optimisationNotification != NULL);
00094
00100     const Duration_T lDuration = iDCPDate - DEFAULT_EVENT_OLDEST_DATETIME;
00101     _eventTimeStamp = lDuration.total_milliseconds();
00102   }
00103
00104   // ////////////////////////////////////////////////////////////////////
00105   EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00106                             SnapshotPtr_T ioSnapshotPtr)
00107     : _eventType (iEventType) {
00108
00109     //
00110     assert (ioSnapshotPtr != NULL);
00111
00112 #if BOOST_VERSION >= 103900
00113     _snapshot = boost::make_shared<SnapshotStruct> (*ioSnapshotPtr);
00114 #else  // BOOST_VERSION >= 103900
00115     _snapshot = ioSnapshotPtr;
00116 #endif // BOOST_VERSION >= 103900
00117     assert (_snapshot != NULL);
00118
00124     const Duration_T lDuration =
00125       _snapshot->getSnapshotTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00126     _eventTimeStamp = lDuration.total_milliseconds();
00127   }
00128
00129   // ////////////////////////////////////////////////////////////////////
00130   EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00131                             RMEventPtr_T ioRMEventPtr)
00132     : _eventType (iEventType) {
00133
00134     //
00135     assert (ioRMEventPtr != NULL);
00136
00137 #if BOOST_VERSION >= 103900
00138     _rmEvent = boost::make_shared<RMEventStruct> (*ioRMEventPtr);
00139 #else  // BOOST_VERSION >= 103900
00140     _rmEvent = ioRMEventPtr;
00141 #endif // BOOST_VERSION >= 103900
00142     assert (_rmEvent != NULL);
00143
00149     const Duration_T lDuration =
00150       _rmEvent->getRMEventTime() - DEFAULT_EVENT_OLDEST_DATETIME;
00151     _eventTimeStamp = lDuration.total_milliseconds();
00152   }
00153
00154   // ////////////////////////////////////////////////////////////////////
00155   EventStruct::EventStruct (const EventType::EN_EventType& iEventType,
00156                             BreakPointPtr_T ioBreakPointPtr)
```

```
    : _eventType (iEventType) {

    //
    assert (ioBreakPointPtr != NULL);

#if BOOST_VERSION >= 103900
    _breakPoint = boost::make_shared<BreakPointStruct> (*ioBreakPointPtr);
#else  // BOOST_VERSION >= 103900
    _breakPoint = ioBreakPointPtr;
#endif // BOOST_VERSION >= 103900
    assert (_breakPoint != NULL);

    const Duration_T lDuration =
      _breakPoint->getBreakPointTime() - DEFAULT_EVENT_OLDEST_DATETIME;
    _eventTimeStamp = lDuration.total_milliseconds();
  }

  // ////////////////////////////////////////////////////////////////////
  EventStruct::EventStruct (const EventStruct& iEventStruct)
    : _eventType (iEventStruct._eventType),
      _eventTimeStamp (iEventStruct._eventTimeStamp) {

    //
    if (iEventStruct._bookingRequest != NULL) {
#if BOOST_VERSION >= 103900
      _bookingRequest =
        boost::make_shared<BookingRequestStruct>(*iEventStruct._bookingRequest);
#else  // BOOST_VERSION >= 103900
      _bookingRequest = iEventStruct._bookingRequest;
#endif // BOOST_VERSION >= 103900
    }

    //
    if (iEventStruct._cancellation != NULL) {
#if BOOST_VERSION >= 103900
      _cancellation =
        boost::make_shared<CancellationStruct>(*iEventStruct._cancellation);
#else  // BOOST_VERSION >= 103900
      _cancellation = iEventStruct._cancellation;
#endif // BOOST_VERSION >= 103900
    }

    //
    if (iEventStruct._optimisationNotification != NULL) {
#if BOOST_VERSION >= 103900
      _optimisationNotification =
        boost::make_shared<OptimisationNotificationStruct> (*iEventStruct._optimisationNotification);
#else  // BOOST_VERSION >= 103900
      _optimisationNotification = iEventStruct._optimisationNotification;
#endif // BOOST_VERSION >= 103900
    }

    //
    if (iEventStruct._snapshot != NULL) {
#if BOOST_VERSION >= 103900
      _snapshot = boost::make_shared<SnapshotStruct> (*iEventStruct._snapshot);
#else  // BOOST_VERSION >= 103900
      _snapshot = iEventStruct._snapshot;
#endif // BOOST_VERSION >= 103900
    }

    //
    if (iEventStruct._rmEvent != NULL) {
#if BOOST_VERSION >= 103900
      _rmEvent = boost::make_shared<RMEventStruct> (*iEventStruct._rmEvent);
#else  // BOOST_VERSION >= 103900
```

```
      _rmEvent = iEventStruct._rmEvent;
#endif // BOOST_VERSION >= 103900
    }

    //
    if (iEventStruct._breakPoint != NULL) {
#if BOOST_VERSION >= 103900
      _breakPoint = boost::make_shared<BreakPointStruct> (*iEventStruct._breakPoint);
#else  // BOOST_VERSION >= 103900
      _breakPoint = iEventStruct._breakPoint;
#endif // BOOST_VERSION >= 103900
    }
  }

  // ////////////////////////////////////////////////////////////////////
  EventStruct::~EventStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  void EventStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string EventStruct::describe() const {
    std::ostringstream oStr;

    //
    const Duration_T lEventDateTimeDelta =
      boost::posix_time::milliseconds (_eventTimeStamp);
    const DateTime_T lEventDateTime (DEFAULT_EVENT_OLDEST_DATETIME
                                     + lEventDateTimeDelta);

    oStr << lEventDateTime;

    //
    switch (_eventType) {
    case EventType::BKG_REQ: {
      assert (_bookingRequest != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
           << ", " << _bookingRequest->describe();
      break;
    }
    case EventType::CX: {
      assert (_cancellation != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
           << ", " << _cancellation->describe();
      break;
    }
    case EventType::OPT_NOT_4_FD: {
      assert (_optimisationNotification != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
           << ", " << _optimisationNotification->describe();
      break;
    }
    case EventType::SNAPSHOT: {
      assert (_snapshot != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
         << ", " << _snapshot->describe();
      break;
    }
    case EventType::RM: {
      assert (_rmEvent != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
         << ", " << _rmEvent->describe();
      break;
    }
```

```
    case EventType::BRK_PT: {
      assert (_breakPoint != NULL);
      oStr << ", " << EventType::getLabel(_eventType)
           << ", " << _breakPoint->describe();
      break;
    }
    default: {
      oStr << ", " << _eventType << " (not yet recognised)";
      break;
    }
    }

    oStr << "\n";
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const DateTime_T& EventStruct::getEventTime() const {
    const DateTime_T& lDateTime (DEFAULT_EVENT_OLDEST_DATETIME);

    //
    switch (_eventType) {
    case EventType::BKG_REQ: {
      assert (_bookingRequest != NULL);
      return _bookingRequest->getRequestDateTime();
      break;
    }
    case EventType::CX: {
      assert (_cancellation != NULL);
      return _cancellation->getCancellationDateTime() ;
      break;
    }
    case EventType::OPT_NOT_4_FD: {
      assert (_optimisationNotification != NULL);
      return _optimisationNotification->getNotificationDateTime();
      break;
    }
    case EventType::SNAPSHOT: {
      assert (_snapshot != NULL);
      return _snapshot->getSnapshotTime();
      break;
    }
    case EventType::RM: {
      assert (_rmEvent != NULL);
      return _rmEvent->getRMEventTime();
      break;
    }
    case EventType::BRK_PT: {
      assert (_breakPoint != NULL);
      return _breakPoint->getBreakPointTime();
      break;
    }
    default: {
      assert(false);
      return lDateTime;
      break;
    }
    }

    return lDateTime;
  }

  // ////////////////////////////////////////////////////////////////////
  void EventStruct::incrementEventTimeStamp() {
    // The date-time is counted in milliseconds (1e-3 second). Hence,
    // one thousand (1e3) of attempts correspond to 1 second.
    // Increment the time stamp of one millisecond.
```

```
    ++_eventTimeStamp;
  }

}
```

33.283 stdair/bom/EventStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
#include <stdair/bom/CancellationTypes.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/bom/BreakPointTypes.hpp>
```

Classes

- struct stdair::EventStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.284 stdair/bom/EventStruct.hpp

```
#ifndef __STDAIR_BAS_EVENTSTRUCT_HPP
#define __STDAIR_BAS_EVENTSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/basic/EventType.hpp>
#include <stdair/bom/EventTypes.hpp>
#include <stdair/bom/BookingRequestTypes.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
#include <stdair/bom/CancellationTypes.hpp>
#include <stdair/bom/RMEventTypes.hpp>
#include <stdair/bom/BreakPointTypes.hpp>

namespace stdair {

  struct EventStruct : public StructAbstract {

    // ////////////// Getters ///////////
  public:
    const EventType::EN_EventType& getEventType() const {
      return _eventType;
    }

    const LongDuration_T& getEventTimeStamp() const {
      return _eventTimeStamp;
    }

    const DateTime_T& getEventTime () const;

    const BookingRequestStruct& getBookingRequest() const {
      assert (_bookingRequest != NULL);
      return *_bookingRequest;
    }

    const CancellationStruct& getCancellation() const {
      assert (_cancellation != NULL);
      return *_cancellation;
    }

    const OptimisationNotificationStruct&
    getOptimisationNotificationStruct() const {
      assert (_optimisationNotification != NULL);
      return *_optimisationNotification;
    }

    const SnapshotStruct& getSnapshotStruct() const {
      assert (_snapshot != NULL);
      return *_snapshot;
    }

    const RMEventStruct& getRMEvent() const {
      assert (_rmEvent != NULL);
      return *_rmEvent;
    }

```

```
00118     const BreakPointStruct& getBreakPoint() const {
00119       assert (_breakPoint != NULL);
00120       return *_breakPoint;
00121     }
00122
00123     // ////////// Display methods //////////
00124   public:
00127     void fromStream (std::istream& ioIn);
00128
00130     const std::string describe() const;
00131
00132
00133     // ////////// Constructors and destructors /////////
00134   public:
00136     EventStruct();
00138     EventStruct (const EventType::EN_EventType&, BookingRequestPtr_T);
00140     EventStruct (const EventType::EN_EventType&, CancellationPtr_T);
00142     EventStruct (const EventType::EN_EventType&, const DateTime_T& iDCPDate,
00143                  OptimisationNotificationPtr_T);
00145     EventStruct (const EventType::EN_EventType&, SnapshotPtr_T);
00147     EventStruct (const EventType::EN_EventType&, RMEventPtr_T);
00149     EventStruct (const EventType::EN_EventType&, BreakPointPtr_T);
00151     EventStruct (const EventStruct&);
00152
00154     ~EventStruct();
00155
00156     // ////////// Modifiers /////////
00157   public:
00166     void incrementEventTimeStamp();
00167
00168     // ////////////////// Attributes //////////////////
00169   private:
00173     EventType::EN_EventType _eventType;
00174
00180     LongDuration_T _eventTimeStamp;
00181
00185     BookingRequestPtr_T _bookingRequest;
00186
00190     CancellationPtr_T _cancellation;
00191
00195     OptimisationNotificationPtr_T _optimisationNotification;
00196
00200     SnapshotPtr_T _snapshot;
00201
00205     RMEventPtr_T _rmEvent;
00206
00210     BreakPointPtr_T _breakPoint;
00211   };
00212
00213 }
00214 #endif // __STDAIR_BAS_EVENTSTRUCT_HPP
```

33.285 stdair/bom/EventTypes.hpp File Reference

```
#include <map>
#include <boost/shared_ptr.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_event_types.hpp>
#include <stdair/basic/ProgressStatus.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::pair< const LongDuration_T, EventStruct > stdair::EventListElement_T
- typedef std::map< const LongDuration_T, EventStruct > stdair::EventList_T

33.286 stdair/bom/EventTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_EVENTTYPES_HPP
00003 #define __STDAIR_BOM_EVENTTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 // Boost Smart Pointers
00011 #include <boost/shared_ptr.hpp>
00012 // StdAir
00013 #include <stdair/stdair_basic_types.hpp>
00014 #include <stdair/stdair_date_time_types.hpp>
00015 #include <stdair/stdair_event_types.hpp>
00016 #include <stdair/basic/ProgressStatus.hpp>
00017 #include <stdair/bom/key_types.hpp>
00018
00019 namespace stdair {
00020
00022   struct EventStruct;
00023
00027   typedef std::pair<const LongDuration_T, EventStruct> EventListElement_T;
00028
00032   typedef std::map<const LongDuration_T, EventStruct> EventList_T;
00033 }
00034 #endif // __STDAIR_BOM_EVENTTYPES_HPP
00035
```

33.287 stdair/bom/FareFamily.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.288 stdair/bom/FareFamily.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamily.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FareFamily::FareFamily() : _key (DEFAULT_FARE_FAMILY_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  FareFamily::FareFamily (const FareFamily& iFareFamily)
    : _key (iFareFamily._key),
      _parent (NULL),
      _frat5Curve (iFareFamily._frat5Curve),
      _disutilityCurve (iFareFamily._disutilityCurve),
      _meanStdDev (iFareFamily._meanStdDev) {
  }

  // ////////////////////////////////////////////////////////////////////
  FareFamily::FareFamily (const Key_T& iKey) : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  FareFamily::~FareFamily() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string FareFamily::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void FareFamily::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void FareFamily::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void FareFamily::serialize (Archive& ioArchive,
                              const unsigned int iFileVersion) {
    ioArchive & _key;
  }
```

```
00066
00067 }
00068
00069
```

33.289 stdair/bom/FareFamily.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>
```

Classes

- class stdair::FareFamily

  Class representing the actual attributes for a family fare.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.290 stdair/bom/FareFamily.hpp

```
#ifndef __STDAIR_BOM_FAREFAMILY_HPP
#define __STDAIR_BOM_FAREFAMILY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
#include <stdair/bom/FareFamilyTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  class FareFamily : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef FareFamilyKey Key_T;


  public:
    // /////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const FamilyCode_T& getFamilyCode() const {
      return _key.getFamilyCode();
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const FRAT5Curve_T& getFrat5Curve () const {
      return _frat5Curve;
    }

    const FFDisutilityCurve_T& getDisutilityCurve () const {
      return _disutilityCurve;
    }

    const MeanValue_T& getMean () const { return _mean; }
    const StdDevValue_T& getStdDev () const {return _stdDev; }

    const MeanStdDevPairVector_T& getMeanStdDev () const {return _meanStdDev; }

```

```
00081
00082   public:
00083     // /////////// Setters ////////////
00085     void setFrat5Curve (const FRAT5Curve_T& iFRAT5Curve) {
00086       _frat5Curve = iFRAT5Curve;
00087     }
00088
00090     void setDisutilityCurve (const FFDisutilityCurve_T& iDisutilityCurve) {
00091       _disutilityCurve = iDisutilityCurve;
00092     }
00093
00095     void setMean (const MeanValue_T& iMean) { _mean = iMean; }
00096     void setStdDev (const StdDevValue_T& iStdDev) { _stdDev = iStdDev; }
00097
00099     void setMeanStdDev (const MeanStdDevPairVector_T& iMeanStdDev){
00100       _meanStdDev = iMeanStdDev;
00101     }
00102
00103
00104   public:
00105     // /////////// Display support methods /////////
00111     void toStream (std::ostream& ioOut) const {
00112       ioOut << toString();
00113     }
00114
00120     void fromStream (std::istream& ioIn) {
00121     }
00122
00126     std::string toString() const;
00127
00131     const std::string describeKey() const {
00132       return _key.toString();
00133     }
00134
00135
00136   public:
00137     // /////////// (Boost) Serialisation support methods /////////
00141     template<class Archive>
00142     void serialize (Archive& ar, const unsigned int iFileVersion);
00143
00144
00145   private:
00150     void serialisationImplementationExport() const;
00151     void serialisationImplementationImport();
00152
00153
00154   protected:
00155     // ////////// Constructors and destructors /////////
00159     FareFamily (const Key_T&);
00160
00164     virtual ~FareFamily();
00165
00166
00167   private:
00171     FareFamily();
00172
00176     FareFamily (const FareFamily&);
00177
00178
00179   public:
00180     // ////////// Attributes /////////
00184     Key_T _key;
00185
00189     BomAbstract* _parent;
00190
00194     HolderMap_T _holderMap;
00195
```

```
    FRAT5Curve_T _frat5Curve;

    FFDisutilityCurve_T _disutilityCurve;

    MeanValue_T _mean;
    StdDevValue_T _stdDev;

    MeanStdDevPairVector_T _meanStdDev;
  };

}
#endif // __STDAIR_BOM_FAREFAMILY_HPP

```

33.291 stdair/bom/FareFamilyKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/FareFamilyKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::FareFamilyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FareFamilyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.292 stdair/bom/FareFamilyKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/FareFamilyKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   FareFamilyKey::FareFamilyKey() : _familyCode (DEFAULT_FARE_FAMILY_CODE) {
00019     assert (false);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   FareFamilyKey::FareFamilyKey (const FareFamilyKey& iFareFamilyKey)
00024     : _familyCode (iFareFamilyKey._familyCode) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   FareFamilyKey::FareFamilyKey (const FamilyCode_T& iFamilyCode)
00029     : _familyCode (iFamilyCode) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   FareFamilyKey::~FareFamilyKey() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void FareFamilyKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "FareFamilyKey: " << toString();
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void FareFamilyKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string FareFamilyKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _familyCode;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void FareFamilyKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void FareFamilyKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void FareFamilyKey::serialize (Archive& ioArchive,
00069                                  const unsigned int iFileVersion) {
00074     ioArchive & _familyCode;
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   // Explicit template instantiation
00079   namespace ba = boost::archive;
00080   template void FareFamilyKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00081                                                              unsigned int);
00082   template void FareFamilyKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00083                                                              unsigned int);
00084   // ////////////////////////////////////////////////////////////////////
00085
00086 }
```

33.293 stdair/bom/FareFamilyKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::FareFamilyKey

  Key of a given fare family, made of a fare family code.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.294 stdair/bom/FareFamilyKey.hpp

```
00001 #ifndef __STDAIR_BOM_FAREFAMILYKEY_HPP
00002 #define __STDAIR_BOM_FAREFAMILYKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct FareFamilyKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     FareFamilyKey();
00035
00036   public:
00040     FareFamilyKey (const FamilyCode_T& iFamilyCode);
00041
00045     FareFamilyKey (const FareFamilyKey&);
00046
00050     ~FareFamilyKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const FamilyCode_T& getFamilyCode () const {
00057       return _familyCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     FamilyCode_T _familyCode;
```

```
00112   };
00113
00114 }
00115 #endif // __STDAIR_BOM_FAREFAMILYKEY_HPP
```

33.295 stdair/bom/FareFamilyTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< FareFamily * > stdair::FareFamilyList_T
- typedef std::map< const MapKey_T, FareFamily * > stdair::FareFamilyMap_T

33.296 stdair/bom/FareFamilyTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_FAREFAMILYTYPES_HPP
#define __STDAIR_BOM_FAREFAMILYTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class FareFamily;

  typedef std::list<FareFamily*> FareFamilyList_T;

  typedef std::map<const MapKey_T, FareFamily*> FareFamilyMap_T;
}
#endif // __STDAIR_BOM_FAREFAMILYTYPES_HPP
```

33.297 stdair/bom/FareFeatures.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FareFeatures.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.298 stdair/bom/FareFeatures.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FareFeatures.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FareFeatures::FareFeatures()
    : _key (TRIP_TYPE_ONE_WAY,
            NO_ADVANCE_PURCHASE,
            SATURDAY_STAY,
            CHANGE_FEES,
            NON_REFUNDABLE,
            NO_STAY_DURATION),
      _parent (NULL)  {
    // That constructor is used by the serialisation process
  }

  // ////////////////////////////////////////////////////////////////////
  FareFeatures::FareFeatures (const FareFeatures& iFeatures)
    : _key (iFeatures.getKey()), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  FareFeatures::FareFeatures (const Key_T& iKey)
    : _key (iKey), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  FareFeatures::~FareFeatures () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string FareFeatures::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool FareFeatures::
  isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
    bool oIsTripTypeValidFlag = true;

    const TripType_T& lFareTripType = getTripType();
    // Check whether the fare trip type is the same as the booking request
    // trip type
    if (iBookingRequestTripType == lFareTripType) {
      // One way case
      return oIsTripTypeValidFlag;
    }

    if (iBookingRequestTripType == TRIP_TYPE_INBOUND
        || iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
      // Round trip case
      if (lFareTripType == TRIP_TYPE_ROUND_TRIP) {
        return oIsTripTypeValidFlag;
```

```
      }
    }

    oIsTripTypeValidFlag = false;
    return oIsTripTypeValidFlag;
  }

  // ////////////////////////////////////////////////////////////////////
  bool FareFeatures::
  isStayDurationValid (const DayDuration_T& iStayDuration) const {

    // Check if the stay duration is lower or equal to the minimum one.
    const DayDuration_T& lMinimumDayDuration = getMinimumStay();
    if (lMinimumDayDuration > iStayDuration) {
      return false;
    }

    return true;
  }

  // ////////////////////////////////////////////////////////////////////
  bool FareFeatures::
  isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
                          const DateTime_T& iFlightDateTime) const {
    bool oIsAdvancePurchaseValidFlag = true;

    // Check whether the departure date is within the date range.
    const DayDuration_T& lAdvancedPurchase = getAdvancePurchase();
    const DateOffset_T lMinimumAdvancedPurchase (lAdvancedPurchase);
    const DateTime_T lCriticalDate = iFlightDateTime - lMinimumAdvancedPurchase;

    if (lCriticalDate < iBookingRequestDateTime) {
      oIsAdvancePurchaseValidFlag = false;
      return oIsAdvancePurchaseValidFlag;
    }

    return true;
  }

}
```

33.299 stdair/bom/FareFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
#include <stdair/bom/FareFeaturesTypes.hpp>
```

Classes

- class stdair::FareFeatures

  Class representing the actual attributes for a fare date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.300 stdair/bom/FareFeatures.hpp

```
00001 #ifndef __STDAIR_BOM_FAREFEATURES_HPP
00002 #define __STDAIR_BOM_FAREFEATURES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FareFeaturesKey.hpp>
00010 #include <stdair/bom/FareFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018   class FareFeatures : public BomAbstract {
00019     template <typename BOM> friend class FacBom;
00020     template <typename BOM> friend class FacCloneBom;
00021     friend class FacBomManager;
00022
00023   public:
00024     // ////////////// Type definitions //////////////
00028     typedef FareFeaturesKey Key_T;
00029
00030   public:
00031     // /////////// Display support methods /////////
00037     void toStream (std::ostream& ioOut) const {
00038       ioOut << toString();
00039     }
00040
00046     void fromStream (std::istream& ioIn) {
00047     }
00048
00052     std::string toString() const;
00053
00057     const std::string describeKey() const {
00058       return _key.toString();
00059     }
00060
00061
00062   public:
00063     // ////////// Getters ////////////
00067     const Key_T& getKey() const {
00068       return _key;
00069     }
00070
00074     BomAbstract* const getParent() const {
00075       return _parent;
00076     }
00077
00081     const HolderMap_T& getHolderMap() const {
00082       return _holderMap;
00083     }
00084
00088     const TripType_T& getTripType() const {
00089       return _key.getTripType();
00090     }
00091
00095     const DayDuration_T& getAdvancePurchase() const {
00096       return _key.getAdvancePurchase();
00097     }
00098
00102     const SaturdayStay_T& getSaturdayStay() const {
00103       return _key.getSaturdayStay();
00104     }
00105
```

```
    const ChangeFees_T& getChangeFees() const {
      return _key.getChangeFees();
    }

    const NonRefundable_T& getRefundableOption() const {
      return _key.getRefundableOption();
    }

    const DayDuration_T& getMinimumStay() const {
      return _key.getMinimumStay();
    }


  public:
    // ############## Business methods ###############
    bool isTripTypeValid (const TripType_T&) const;

    bool isStayDurationValid (const DayDuration_T&) const;

    bool isAdvancePurchaseValid (const DateTime_T& iBookingRequestDateTime,
                                 const DateTime_T& iFlightDateTime) const;


  protected:
    // ########## Constructors and destructors #########
    FareFeatures (const Key_T&);
    virtual ~FareFeatures ();

  private:
    FareFeatures ();
    FareFeatures (const FareFeatures&);

  protected:
    // ############# Attributes #############
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;
  };

}
#endif // __STDAIR_BOM_FAREFEATURES_HPP
```

33.301 stdair/bom/FareFeaturesKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_DefaultObject.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/FareFeaturesKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.302 stdair/bom/FareFeaturesKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <ostream>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_DefaultObject.hpp>
00009 #include <stdair/basic/BasConst_Request.hpp>
00010 #include <stdair/bom/FareFeaturesKey.hpp>
00011
00012 namespace stdair {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   FareFeaturesKey::FareFeaturesKey()
00016     : _tripType (TRIP_TYPE_ONE_WAY),
00017       _advancePurchase (NO_ADVANCE_PURCHASE),
00018       _saturdayStay (SATURDAY_STAY),
00019       _changeFees (CHANGE_FEES),
00020       _nonRefundable (NON_REFUNDABLE),
00021       _minimumStay (NO_STAY_DURATION) {
00022     assert (false);
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   FareFeaturesKey::FareFeaturesKey (const TripType_T& iTripType,
00027                                     const DayDuration_T& iAdvancePurchase,
00028                                     const SaturdayStay_T& iSaturdayStay,
00029                                     const ChangeFees_T& iChangeFees,
00030                                     const NonRefundable_T& iNonRefundable,
00031                                     const DayDuration_T& iMinimumStay)
00032     : _tripType (iTripType), _advancePurchase (iAdvancePurchase),
00033       _saturdayStay (iSaturdayStay), _changeFees (iChangeFees),
00034       _nonRefundable (iNonRefundable), _minimumStay (iMinimumStay) {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   FareFeaturesKey::FareFeaturesKey (const FareFeaturesKey& iKey)
00039     : _tripType (iKey.getTripType()),
00040       _advancePurchase (iKey.getAdvancePurchase()),
00041       _saturdayStay (iKey.getSaturdayStay()),
00042       _changeFees (iKey.getChangeFees()),
00043       _nonRefundable (iKey.getRefundableOption()),
00044       _minimumStay (iKey.getMinimumStay()) {
00045   }
00046
00047   // ////////////////////////////////////////////////////////////////////
00048   FareFeaturesKey::~FareFeaturesKey() {
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   void FareFeaturesKey::toStream (std::ostream& ioOut) const {
00053     ioOut << "FareFeaturesKey: " << toString() << std::endl;
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   void FareFeaturesKey::fromStream (std::istream& ioIn) {
00058   }
00059
00060   // ////////////////////////////////////////////////////////////////////
00061   const std::string FareFeaturesKey::toString() const {
00062     std::ostringstream oStr;
00063     oStr << _tripType << " -- " << _advancePurchase << "-"
00064          << _saturdayStay << "-" << _changeFees << "-"
00065          << _nonRefundable << "-" << _minimumStay;
```

```
    return oStr.str();
  }

}
```

33.303 stdair/bom/FareFeaturesKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Classes

- struct stdair::FareFeaturesKey

  Key of date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.304 stdair/bom/FareFeaturesKey.hpp

```
00001 #ifndef __STDAIR_BOM_FAREFEATURESKEY_HPP
00002 #define __STDAIR_BOM_FAREFEATURESKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012
00013 namespace stdair  {
00014
00018   struct FareFeaturesKey : public KeyAbstract {
00019   public:
00020     // /////////// Construction ///////////
00022     FareFeaturesKey (const TripType_T&, const DayDuration_T&,
00023                      const SaturdayStay_T&, const ChangeFees_T&,
00024                      const NonRefundable_T&, const DayDuration_T&);
00026     FareFeaturesKey (const FareFeaturesKey&);
00028     ~FareFeaturesKey ();
00029   private:
00031     FareFeaturesKey();
00032
00033
00034   public:
00035     // /////////// Getters //////////
00039     const TripType_T& getTripType() const {
00040       return _tripType;
00041     }
00042
00046     const DayDuration_T& getAdvancePurchase() const {
00047       return _advancePurchase;
00048     }
00049
00053     const SaturdayStay_T& getSaturdayStay() const {
00054       return _saturdayStay;
00055     }
00056
00060     const ChangeFees_T& getChangeFees() const {
00061       return _changeFees;
00062     }
00063
00067     const NonRefundable_T& getRefundableOption() const {
00068       return _nonRefundable;
00069     }
00070
00074     const DayDuration_T& getMinimumStay() const {
00075       return _minimumStay;
00076     }
00077
00078
00079   public:
00080     // /////////// Display support methods /////////
00086     void toStream (std::ostream& ioOut) const;
00087
00093     void fromStream (std::istream& ioIn);
00094
00100     const std::string toString() const;
00101
00102
00103   private:
00104     // ///////////////// Attributes ///////////////////
00108     TripType_T _tripType;
```

```
00109
00113     DayDuration_T _advancePurchase;
00114
00118     SaturdayStay_T _saturdayStay;
00119
00123     ChangeFees_T _changeFees;
00124
00128     NonRefundable_T _nonRefundable;
00129
00133     DayDuration_T _minimumStay;
00134   };
00135
00136 }
00137 #endif // __STDAIR_BOM_FAREFEATURESKEY_HPP
```

33.305 stdair/bom/FareFeaturesTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< FareFeatures * > stdair::FareFeaturesList_T
- typedef std::map< const MapKey_T, FareFeatures * > stdair::FareFeaturesMap_T
- typedef std::pair< MapKey_T, FareFeatures * > stdair::FareFeaturesWithKey_T
- typedef std::list< FareFeaturesWithKey_T > stdair::FareFeaturesDetailedList_T

33.306 stdair/bom/FareFeaturesTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_FAREFEATURESTYPES_HPP
#define __STDAIR_BOM_FAREFEATURESTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class FareFeatures;

  typedef std::list<FareFeatures*> FareFeaturesList_T;

  typedef std::map<const MapKey_T, FareFeatures*> FareFeaturesMap_T;

  typedef std::pair<MapKey_T, FareFeatures*> FareFeaturesWithKey_T;
  typedef std::list<FareFeaturesWithKey_T> FareFeaturesDetailedList_T;
}
#endif // __STDAIR_BOM_FAREFEATURESTYPES_HPP

```

33.307 stdair/bom/FareOptionStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.308 stdair/bom/FareOptionStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/FareOptionStruct.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   FareOptionStruct::FareOptionStruct()
00015     : _fare (DEFAULT_FARE_VALUE), _avl (DEFAULT_AVAILABILITY) {
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   FareOptionStruct::FareOptionStruct (const FareOptionStruct& iFO)
00020     : _classPath (iFO._classPath),
00021       _fare (iFO._fare), _avl (iFO._avl), _changeFee (iFO._changeFee),
00022       _nonRefundable (iFO._nonRefundable), _saturdayStay (iFO._saturdayStay) {
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   FareOptionStruct::FareOptionStruct (const std::string& iClassPath,
00027                                       const Fare_T& iFare,
00028                                       const ChangeFees_T& iChangeFee,
00029                                       const NonRefundable_T& iNonRefundable,
00030                                       const SaturdayStay_T& iSaturdayNightStay)
00031     : _fare (iFare), _avl (DEFAULT_AVAILABILITY),
00032       _changeFee (iChangeFee), _nonRefundable (iNonRefundable),
00033       _saturdayStay (iSaturdayNightStay) {
00034     _classPath.push_back (iClassPath);
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   FareOptionStruct::~FareOptionStruct() {
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void FareOptionStruct::toStream (std::ostream& ioOut) const {
00043     ioOut << describe();
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   void FareOptionStruct::fromStream (std::istream& ioIn) {
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   const std::string FareOptionStruct::describe() const {
00052     std::ostringstream oStr;
00053
00054     oStr  << "Class path: ";
00055     unsigned short idx = 0;
00056     for (ClassList_StringList_T::const_iterator itClassPath =
00057            _classPath.begin(); itClassPath != _classPath.end();
00058          ++itClassPath, ++idx) {
00059       if (idx != 0) {
00060         oStr << "-";
00061       }
00062       const std::string& lClassPath = *itClassPath;
00063       oStr << lClassPath;
00064     }
00065
```

```
    oStr << "; " << _fare << " EUR";
    oStr << "; conditions: " << _changeFee  << " " << _nonRefundable
         << " " << _saturdayStay;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string FareOptionStruct::display() const {
    std::ostringstream oStr;

    unsigned short idx = 0;
    for (ClassList_StringList_T::const_iterator itClassPath =
           _classPath.begin(); itClassPath != _classPath.end();
         ++itClassPath, ++idx) {
      if (idx != 0) {
        oStr << "-";
      }
      const std::string& lClassPath = *itClassPath;
      oStr << lClassPath;
    }

    oStr << ", " << _fare << ", " << _changeFee  << " " << _nonRefundable
         << " " << _saturdayStay;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void FareOptionStruct::addClassList (const std::string iClassCodeList) {
    _classPath.push_back (iClassCodeList);
  }

  // ////////////////////////////////////////////////////////////////////
  void FareOptionStruct::emptyClassList () {
    _classPath.clear();
  }

}
```

33.309 stdair/bom/FareOptionStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Classes

- struct stdair::FareOptionStruct

  Structure holding the elements of a fare option.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.310 stdair/bom/FareOptionStruct.hpp

```
#ifndef __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
#define __STDAIR_BOM_FAREOPTIONSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

namespace stdair {

  struct FareOptionStruct : public StructAbstract {
  public:
    // /////////// Getters ///////////////
    const ClassList_StringList_T& getClassPath() const {
      return _classPath;
    }

    const Fare_T& getFare() const {
      return _fare;
    }

    const Availability_T& getAvailability() const {
      return _avl;
    }

    const ChangeFees_T getChangeFees() const {
      return _changeFee;
    }

    const NonRefundable_T getNonRefundable() const {
      return _nonRefundable;
    }

    const SaturdayStay_T getSaturdayStay() const {
      return _saturdayStay;
    }


  public:
    // /////////// Setters ///////////////
    void addClassList (const std::string);

    void emptyClassList ();

    void setFare (const Fare_T& iFare) {
      _fare = iFare;
    }

    void setAvailability (const Availability_T& iAvl) {
      _avl = iAvl;
    }

    void setChangeFees (const ChangeFees_T iRes) {
      _changeFee = iRes;
    }

    void setNonRefundable (const NonRefundable_T iRes) {
      _nonRefundable = iRes;
    }
```

```
00081
00083     void setSaturdayStay (const SaturdayStay_T iRes) {
00084       _saturdayStay = iRes;
00085     }
00086
00087
00088   public:
00089     // /////////// Display support method /////////////
00095     void toStream (std::ostream& ioOut) const;
00096
00102     void fromStream (std::istream& ioIn);
00103
00107     const std::string describe() const;
00108
00112     const std::string display() const;
00113
00114
00115   public:
00116     // ///////////// Constructors & Destructor ///////////////
00120     FareOptionStruct();
00121
00125     FareOptionStruct (const std::string& iClassPath,
00126                       const Fare_T&, const ChangeFees_T&,
00127                       const NonRefundable_T&, const SaturdayStay_T&);
00128
00132     FareOptionStruct (const FareOptionStruct&);
00133
00137     ~FareOptionStruct();
00138
00139
00140   private:
00141     // /////////////////////// Attributes /////////////////////
00145     ClassList_StringList_T _classPath;
00146
00150     Fare_T _fare;
00151
00155     Availability_T _avl;
00156
00160     ChangeFees_T _changeFee;
00161
00165     NonRefundable_T _nonRefundable;
00166
00170     SaturdayStay_T _saturdayStay;
00171   };
00172
00173 }
00174 #endif // __STDAIR_BOM_FAREOPTIONSTRUCT_HPP
```

33.311 stdair/bom/FareOptionTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_types.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< FareOptionStruct > stdair::FareOptionList_T

33.312 stdair/bom/FareOptionTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_FAREOPTIONTYPES_HPP
#define __STDAIR_BOM_FAREOPTIONTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <list>
#include <map>
// STDAIR
#include <stdair/stdair_types.hpp>
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  struct FareOptionStruct;

  typedef std::list<FareOptionStruct> FareOptionList_T;

}
#endif // __STDAIR_BOM_FAREOPTIONTYPES_HPP

```

33.313 stdair/bom/FFDisutilityCurveHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.314 stdair/bom/FFDisutilityCurveHolderStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/service/Logger.hpp>
00009 #include <stdair/bom/FFDisutilityCurveHolderStruct.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   FFDisutilityCurveHolderStruct::FFDisutilityCurveHolderStruct() {
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   FFDisutilityCurveHolderStruct::
00019   FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct& iHolder)
00020     : _disutilityCurveHolder (iHolder._disutilityCurveHolder) {
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   FFDisutilityCurveHolderStruct::~FFDisutilityCurveHolderStruct() {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   const FFDisutilityCurve_T& FFDisutilityCurveHolderStruct::
00029   getFFDisutilityCurve (const std::string& iKey) const {
00030     FFDisutilityCurveHolder_T::const_iterator itCurve = _disutilityCurveHolder.fi
      nd (iKey);
00031     if (itCurve == _disutilityCurveHolder.end()) {
00032       STDAIR_LOG_DEBUG ("Cannot find the FFDisutility curve correponding to the "

00033                         << "given key: " << iKey);
00034       assert (false);
00035     }
00036
00037     return itCurve->second;
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   void FFDisutilityCurveHolderStruct::
00042   addCurve (const std::string& iKey, const FFDisutilityCurve_T& iCurve) {
00043     bool insert = _disutilityCurveHolder.insert (FFDisutilityCurveHolder_T::value
      _type(iKey, iCurve)).second;
00044     if (insert == false) {
00045       STDAIR_LOG_DEBUG ("Cannot add the FFDisutility curve correponding to the "
00046                         << "given key: " << iKey
00047                         << ", the key may already exist.");
00048       assert (false);
00049     }
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void FFDisutilityCurveHolderStruct::toStream (std::ostream& ioOut) const {
00054     ioOut << describe();
00055   }
00056
00057   // ////////////////////////////////////////////////////////////////////
00058   void FFDisutilityCurveHolderStruct::fromStream (std::istream& ioIn) {
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   const std::string FFDisutilityCurveHolderStruct::describe() const {
```

```
    std::ostringstream oStr;
    for (FFDisutilityCurveHolder_T::const_iterator itCurve = _disutilityCurveHolder.begin();
         itCurve != _disutilityCurveHolder.end(); ++itCurve) {
      const std::string& lKey = itCurve->first;
      const FFDisutilityCurve_T& lCurve = itCurve->second;
      oStr << lKey << "; ";
      for (FFDisutilityCurve_T::const_reverse_iterator itFFDisutility =
             lCurve.rbegin();itFFDisutility != lCurve.rend(); ++itFFDisutility){
        const DTD_T& lDTD = itFFDisutility->first;
        const double& lFFDisutility = itFFDisutility->second;
        oStr << lDTD << ":" << lFFDisutility << ";";
      }
      oStr << std::endl;
    }
    return oStr.str();
  }

}
```

33.315 stdair/bom/FFDisutilityCurveHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::FFDisutilityCurveHolderStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::map< const std::string, FFDisutilityCurve_T > stdair::FFDisutilityCurveHolder_T

33.316 stdair/bom/FFDisutilityCurveHolderStruct.hpp

```
#ifndef __STDAIR_BOM_FFDISUTILITYCURVEHOLDERSTRUCT_HPP
#define __STDAIR_BOM_FFDISUTILITYCURVEHOLDERSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {
  // Type definition for the holder of disutility curves.
  typedef std::map<const std::string, FFDisutilityCurve_T>
FFDisutilityCurveHolder_T;

  struct FFDisutilityCurveHolderStruct : public StructAbstract {
  public:
    // /////////////// Getters /////////////////
    const FFDisutilityCurve_T& getFFDisutilityCurve (const std::string&) const;

    // ///////////// Business Methods //////////
    void addCurve (const std::string&, const FFDisutilityCurve_T&);

    // /////////// Display support method /////////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string describe() const;


    // /////////////// Constructors and Destructors /////////////////
  public:
    FFDisutilityCurveHolderStruct ();

    FFDisutilityCurveHolderStruct (const FFDisutilityCurveHolderStruct&);

  public:
    ~FFDisutilityCurveHolderStruct();


  private:
    // /////////////// Attributes /////////////////
    FFDisutilityCurveHolder_T _disutilityCurveHolder;
  };

}
#endif // __STDAIR_BOM_FFDISUTILITYCURVEHOLDERSTRUCT_HPP
```

33.317 stdair/bom/FlightDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.318 stdair/bom/FlightDate.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012 #include <stdair/bom/LegDate.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   FlightDate::FlightDate()
00019     : _key (DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE), _parent (NULL) {
00020     // That constructor is used by the serialisation process
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   FlightDate::FlightDate (const FlightDate& iFlightDate)
00025     : _key (iFlightDate._key), _parent (NULL) {
00026   }
00027
00028   // ////////////////////////////////////////////////////////////////////
00029   FlightDate::FlightDate (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   FlightDate::~FlightDate() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   const AirlineCode_T& FlightDate::getAirlineCode() const {
00038     const Inventory* lInventory_ptr =
00039       static_cast<const Inventory*> (getParent());
00040     assert (lInventory_ptr != NULL);
00041     return lInventory_ptr->getAirlineCode();
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   std::string FlightDate::toString() const {
00046     std::ostringstream oStr;
00047     oStr << describeKey();
00048     return oStr.str();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   LegDate* FlightDate::getLegDate (const std::string& iLegDateKeyStr) const {
00053     LegDate* oLegDate_ptr =
00054       BomManager::getObjectPtr<LegDate> (*this, iLegDateKeyStr);
00055     return oLegDate_ptr;
00056   }
00057
00058   // ////////////////////////////////////////////////////////////////////
00059   LegDate* FlightDate::getLegDate (const LegDateKey& iLegDateKey) const {
00060     return getLegDate (iLegDateKey.toString());
00061   }
00062
00063   // ////////////////////////////////////////////////////////////////////
00064   SegmentDate* FlightDate::
00065   getSegmentDate (const std::string& iSegmentDateKeyStr) const {
```

```
    SegmentDate* oSegmentDate_ptr =
      BomManager::getObjectPtr<SegmentDate> (*this, iSegmentDateKeyStr);
    return oSegmentDate_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDate* FlightDate::
  getSegmentDate (const SegmentDateKey& iSegmentDateKey) const {
    return getSegmentDate (iSegmentDateKey.toString());
  }

}
```

33.319 stdair/bom/FlightDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/FlightDateTypes.hpp>
```

Classes

- class stdair::FlightDate

  Class representing the actual attributes for an airline flight-date.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.320 stdair/bom/FlightDate.hpp

```
#ifndef __STDAIR_BOM_FLIGHTDATE_HPP
#define __STDAIR_BOM_FLIGHTDATE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/FlightDateTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct LegDateKey;
  class LegDate;
  struct SegmentDateKey;
  class SegmentDate;

  class FlightDate : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef FlightDateKey Key_T;


  public:
    // /////////// Getters ///////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const FlightNumber_T& getFlightNumber() const {
      return _key.getFlightNumber();
    }

    const Date_T& getDepartureDate() const {
      return _key.getDepartureDate();
    }

    const AirlineCode_T& getAirlineCode() const;

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    LegDate* getLegDate (const std::string& iLegDateKeyStr) const;

```

```
    LegDate* getLegDate (const LegDateKey&) const;

    SegmentDate* getSegmentDate (const std::string& iSegmentDateKeyStr) const;

    SegmentDate* getSegmentDate (const SegmentDateKey&) const;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  protected:
    // /////////// Constructors and destructors /////////
    FlightDate (const Key_T&);

    virtual ~FlightDate();

  private:
    FlightDate();

    FlightDate (const FlightDate&);


  protected:
    // /////////// Attributes /////////
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;
  };

}
#endif // __STDAIR_BOM_FLIGHTDATE_HPP
```

33.321 stdair/bom/FlightDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/FlightDateKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::FlightDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FlightDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.322 stdair/bom/FlightDateKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost Date-Time
#include <boost/date_time/gregorian/formatters.hpp>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/FlightDateKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FlightDateKey::FlightDateKey()
    : _flightNumber (DEFAULT_FLIGHT_NUMBER),
      _departureDate (DEFAULT_DEPARTURE_DATE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  FlightDateKey::FlightDateKey (const FlightNumber_T& iFlightNumber,
                                const Date_T& iFlightDate)
    : _flightNumber (iFlightNumber), _departureDate (iFlightDate) {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightDateKey::FlightDateKey (const FlightDateKey& iKey)
    : _flightNumber (iKey._flightNumber), _departureDate (iKey._departureDate) {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightDateKey::~FlightDateKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightDateKey::toStream (std::ostream& ioOut) const {
    ioOut << "FlightDateKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightDateKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string FlightDateKey::toString() const {
    std::ostringstream oStr;
    const std::string& lDepartureDateStr =
      boost::gregorian::to_iso_extended_string (_departureDate);
    oStr << _flightNumber
         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << lDepartureDateStr;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightDateKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightDateKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void FlightDateKey::serialize (Archive& ioArchive,
                                 const unsigned int iFileVersion) {
    std::string lDepartureDateStr =
      boost::gregorian::to_simple_string (_departureDate);
    ioArchive & _flightNumber & lDepartureDateStr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template void FlightDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                             unsigned int);
  template void FlightDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                             unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

33.323 stdair/bom/FlightDateKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::FlightDateKey

  Key of a given flight-date, made of a flight number and a departure date.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.324 stdair/bom/FlightDateKey.hpp

```
#ifndef __STDAIR_BOM_FLIGHTDATEKEY_HPP
#define __STDAIR_BOM_FLIGHTDATEKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct FlightDateKey : public KeyAbstract {
    friend class boost::serialization::access;

    // /////////// Constructors and destructors ///////////
  private:
    FlightDateKey();

  public:
    FlightDateKey (const FlightNumber_T&, const Date_T&);

    FlightDateKey (const FlightDateKey&);

    ~FlightDateKey();


  public:
    // /////////// Getters //////////
    const FlightNumber_T& getFlightNumber() const {
      return _flightNumber;
    }

    const Date_T& getDepartureDate() const {
      return _departureDate;
    }


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();
```

```
00111
00112
00113   private:
00114     // ///////////////// Attributes ///////////////
00118     FlightNumber_T _flightNumber;
00119
00123     Date_T _departureDate;
00124   };
00125
00126 }
00127 #endif // __STDAIR_BOM_FLIGHTDATEKEY_HPP
```

33.325 stdair/bom/FlightDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< FlightDate * > stdair::FlightDateList_T
- typedef std::map< const MapKey_T, FlightDate * > stdair::FlightDateMap_T

33.326 stdair/bom/FlightDateTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_FLIGHTDATETYPES_HPP
#define __STDAIR_BOM_FLIGHTDATETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations
  class FlightDate;

  // ////////////////////////// Type definitions //////////////////////////
  typedef std::list<FlightDate*> FlightDateList_T;

  typedef std::map<const MapKey_T, FlightDate*> FlightDateMap_T;

}
#endif // __STDAIR_BOM_FLIGHTDATETYPES_HPP

```

33.327 stdair/bom/FlightPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/bom/FlightPeriod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.328 stdair/bom/FlightPeriod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// STDAIR
#include <stdair/bom/FlightPeriod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FlightPeriod::FlightPeriod (const Key_T& iKey)
    : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriod::FlightPeriod (const FlightPeriod& iFlightPeriod)
    : _key (iFlightPeriod.getKey()), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriod::~FlightPeriod () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string FlightPeriod::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

}

```

33.329 stdair/bom/FlightPeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/FlightPeriodKey.hpp>
#include <stdair/bom/FlightPeriodTypes.hpp>
```

Classes

- class stdair::FlightPeriod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.330 stdair/bom/FlightPeriod.hpp

```
00001 #ifndef __STDAIR_BOM_FLIGHTPERIOD_HPP
00002 #define __STDAIR_BOM_FLIGHTPERIOD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/FlightPeriodKey.hpp>
00010 #include <stdair/bom/FlightPeriodTypes.hpp>
00011
00012 namespace stdair {
00013
00015   class FlightPeriod : public BomAbstract {
00016     template <typename BOM> friend class FacBom;
00017     template <typename BOM> friend class FacCloneBom;
00018     friend class FacBomManager;
00019
00020   public:
00021     // Type definitions.
00023     typedef FlightPeriodKey Key_T;
00024
00025   public:
00026     // /////////// Getters ///////////////
00028     const Key_T& getKey () const { return _key; }
00029
00031     BomAbstract* const getParent() const { return _parent; }
00032
00034     const FlightNumber_T& getFlightNumber () const {
00035       return _key.getFlightNumber();
00036     }
00037
00039     const PeriodStruct& getPeriod () const { return _key.getPeriod(); }
00040
00042     const HolderMap_T& getHolderMap() const { return _holderMap; }
00043
00044
00045   public:
00046     // /////////// Display support methods /////////
00049     void toStream (std::ostream& ioOut) const { ioOut << toString(); }
00050
00053     void fromStream (std::istream& ioIn) { }
00054
00056     std::string toString() const;
00057
00059     const std::string describeKey() const { return _key.toString(); }
00060
00061   protected:
00065     FlightPeriod (const Key_T&);
00066
00070     ~FlightPeriod ();
00071
00072   private:
00073
00077     FlightPeriod ();
00078
00082     FlightPeriod (const FlightPeriod&);
00083
00084   protected:
00085     // Attributes
00086     Key_T _key;
00087     BomAbstract* _parent;
00088     HolderMap_T _holderMap;
00089   };
00090
```

```
}
#endif // __STDAIR_BOM_FLIGHTPERIOD_HPP

```

33.331 stdair/bom/FlightPeriodKey.cpp File Reference

```
#include <stdair/bom/FlightPeriodKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.332 stdair/bom/FlightPeriodKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/FlightPeriodKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FlightPeriodKey::FlightPeriodKey (const FlightNumber_T& iFlightNumber,
                                    const PeriodStruct& iPeriod)
    : _flightNumber (iFlightNumber), _period (iPeriod) {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriodKey::FlightPeriodKey (const FlightPeriodKey& iKey)
    : _flightNumber (iKey._flightNumber), _period (iKey._period) {
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriodKey::~FlightPeriodKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightPeriodKey::toStream (std::ostream& ioOut) const {
    ioOut << "FlightPeriodKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void FlightPeriodKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string FlightPeriodKey::toString() const {
    std::ostringstream oStr;
    oStr << _flightNumber << ", " << _period.describeShort();
    return oStr.str();
  }

}
```

33.333 stdair/bom/FlightPeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>
```

Classes

- struct stdair::FlightPeriodKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.334 stdair/bom/FlightPeriodKey.hpp

```
#ifndef __STDAIR_BOM_FLIGHTPERIODKEY_HPP
#define __STDAIR_BOM_FLIGHTPERIODKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/bom/PeriodStruct.hpp>

namespace stdair {
  struct FlightPeriodKey : public KeyAbstract {

  private:
    // /////////// Default constructor //////////
    FlightPeriodKey ();
  public:
    // /////////// Construction ///////////
    FlightPeriodKey (const FlightNumber_T&, const PeriodStruct&);
    FlightPeriodKey (const FlightPeriodKey&);
    ~FlightPeriodKey ();

    // /////////// Getters //////////
    const FlightNumber_T& getFlightNumber() const {
      return _flightNumber;
    }

    const PeriodStruct& getPeriod () const {
      return _period;
    }

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

  private:
    // Attributes
    FlightNumber_T _flightNumber;

    PeriodStruct _period;

  };

}
#endif // __STDAIR_BOM_FLIGHTPERIODKEY_HPP
```

33.335 stdair/bom/FlightPeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< FlightPeriod * > stdair::FlightPeriodList_T
- typedef std::map< const MapKey_T, FlightPeriod * > stdair::FlightPeriodMap_T

33.336 stdair/bom/FlightPeriodTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_FLIGHTPERIODTYPES_HPP
#define __STDAIR_BOM_FLIGHTPERIODTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class FlightPeriod;

  typedef std::list<FlightPeriod*> FlightPeriodList_T;

  typedef std::map<const MapKey_T, FlightPeriod*> FlightPeriodMap_T;
}
#endif // __STDAIR_BOM_FLIGHTPERIODTYPES_HPP

```

33.337 stdair/bom/FRAT5CurveHolderStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.338 stdair/bom/FRAT5CurveHolderStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/service/Logger.hpp>
#include <stdair/bom/FRAT5CurveHolderStruct.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FRAT5CurveHolderStruct::FRAT5CurveHolderStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  FRAT5CurveHolderStruct::
  FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct& iHolder)
    : _frat5CurveHolder (iHolder._frat5CurveHolder) {
  }

  // ////////////////////////////////////////////////////////////////////
  FRAT5CurveHolderStruct::~FRAT5CurveHolderStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  const FRAT5Curve_T& FRAT5CurveHolderStruct::
  getFRAT5Curve (const std::string& iKey) const {
    FRAT5CurveHolder_T::const_iterator itCurve = _frat5CurveHolder.find (iKey);
    if (itCurve == _frat5CurveHolder.end()) {
      STDAIR_LOG_DEBUG ("Cannot find the FRAT5 curve correponding to the "
                        << "given key: " << iKey);
      assert (false);
    }

    return itCurve->second;
  }

  // ////////////////////////////////////////////////////////////////////
  void FRAT5CurveHolderStruct::
  addCurve (const std::string& iKey, const FRAT5Curve_T& iCurve) {
    bool insert = _frat5CurveHolder.insert (FRAT5CurveHolder_T::value_type(iKey,
iCurve)).second;
    if (insert == false) {
      STDAIR_LOG_DEBUG ("Cannot add the FRAT5 curve correponding to the "
                        << "given key: " << iKey
                        << ", the key may already exist.");
      assert (false);
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void FRAT5CurveHolderStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void FRAT5CurveHolderStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string FRAT5CurveHolderStruct::describe() const {
    std::ostringstream oStr;
    for (FRAT5CurveHolder_T::const_iterator itCurve = _frat5CurveHolder.begin();
```

```
00065          itCurve != _frat5CurveHolder.end(); ++itCurve) {
00066       const std::string& lKey = itCurve->first;
00067       const FRAT5Curve_T& lCurve = itCurve->second;
00068       oStr << lKey << "; ";
00069       for (FRAT5Curve_T::const_reverse_iterator itFRAT5 = lCurve.rbegin();
00070            itFRAT5 != lCurve.rend(); ++itFRAT5) {
00071         const DTD_T& lDTD = itFRAT5->first;
00072         const double& lFRAT5 = itFRAT5->second;
00073         oStr << lDTD << ":" << lFRAT5 << ";";
00074       }
00075       oStr << std::endl;
00076     }
00077     return oStr.str();
00078   }
00079
00080 }
```

33.339 stdair/bom/FRAT5CurveHolderStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
```

Classes

- struct stdair::FRAT5CurveHolderStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::map< const std::string, FRAT5Curve_T > stdair::FRAT5CurveHolder_T

33.340 stdair/bom/FRAT5CurveHolderStruct.hpp

```
#ifndef __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP
#define __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>

namespace stdair {
  // Type definition for the holder of Frat5 curves.
  typedef std::map<const std::string, FRAT5Curve_T> FRAT5CurveHolder_T;

  struct FRAT5CurveHolderStruct : public StructAbstract {
  public:
    // /////////////// Getters /////////////////
    const FRAT5Curve_T& getFRAT5Curve (const std::string&) const;

    // ///////////// Business Methods //////////
    void addCurve (const std::string&, const FRAT5Curve_T&);

    // /////////// Display support method /////////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string describe() const;


    // /////////////// Constructors and Destructors /////////////////
  public:
    FRAT5CurveHolderStruct ();

    FRAT5CurveHolderStruct (const FRAT5CurveHolderStruct&);

  public:
    ~FRAT5CurveHolderStruct();


  private:
    // /////////////// Attributes /////////////////
    FRAT5CurveHolder_T _frat5CurveHolder;
  };

}
#endif // __STDAIR_BOM_FRAT5CURVEHOLDERSTRUCT_HPP
```

33.341 stdair/bom/Inventory.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.342 stdair/bom/Inventory.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Inventory.hpp>
00009 #include <stdair/bom/BomManager.hpp>
00010 #include <stdair/bom/Inventory.hpp>
00011 #include <stdair/bom/FlightDate.hpp>
00012
00013 namespace stdair {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   Inventory::Inventory() :
00017     _key (DEFAULT_AIRLINE_CODE),
00018     _parent (NULL),
00019     _airlineFeature (NULL) {
00020     // That constructor is used by the serialisation process
00021   }
00022
00023   // ////////////////////////////////////////////////////////////////////
00024   Inventory::Inventory (const Inventory& iInventory)
00025     : _key (iInventory._key),
00026       _parent (NULL),
00027       _airlineFeature (NULL) {
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   Inventory::Inventory (const Key_T& iKey) :
00032     _key (iKey),
00033     _parent (NULL),
00034     _airlineFeature (NULL) {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   Inventory::~Inventory() {
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   std::string Inventory::toString() const {
00043     std::ostringstream oStr;
00044     oStr << describeKey();
00045     return oStr.str();
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   FlightDate* Inventory::
00050   getFlightDate (const std::string& iFlightDateKeyStr) const {
00051     FlightDate* oFlightDate_ptr =
00052       BomManager::getObjectPtr<FlightDate> (*this, iFlightDateKeyStr);
00053     return oFlightDate_ptr;
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   FlightDate* Inventory::
00058   getFlightDate (const FlightDateKey& iFlightDateKey) const {
00059     return getFlightDate (iFlightDateKey.toString());
00060   }
00061
00062   // ////////////////////////////////////////////////////////////////////
00063   ForecastingMethod::EN_ForecastingMethod Inventory::
00064   getForecastingMethod() const {
00065     assert (_airlineFeature != NULL);
```

```
    return _airlineFeature->getForecastingMethod();
  }

  // ////////////////////////////////////////////////////////////////////
  UnconstrainingMethod::EN_UnconstrainingMethod Inventory::
  getUnconstrainingMethod() const {
    assert (_airlineFeature != NULL);
    return _airlineFeature->getUnconstrainingMethod();
  }

  // ////////////////////////////////////////////////////////////////////
  PreOptimisationMethod::EN_PreOptimisationMethod Inventory::
  getPreOptimisationMethod() const {
    assert (_airlineFeature != NULL);
    return _airlineFeature->getPreOptimisationMethod();
  }

  // ////////////////////////////////////////////////////////////////////
  OptimisationMethod::EN_OptimisationMethod Inventory::
  getOptimisationMethod() const {
    assert (_airlineFeature != NULL);
    return _airlineFeature->getOptimisationMethod();
  }

  // ////////////////////////////////////////////////////////////////////
  PartnershipTechnique::EN_PartnershipTechnique Inventory::
  getPartnershipTechnique() const {
    assert (_airlineFeature != NULL);
    return _airlineFeature->getPartnershipTechnique();
  }

}
```

33.343 stdair/bom/Inventory.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/InventoryTypes.hpp>
```

Classes

- class stdair::Inventory

  Class representing the actual attributes for an airline inventory.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.344 stdair/bom/Inventory.hpp

```
#ifndef __STDAIR_BOM_INVENTORY_HPP
#define __STDAIR_BOM_INVENTORY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/InventoryTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct FlightDateKey;
  class FlightDate;

  class Inventory : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public :
    // ////////// Type definitions ////////////
    typedef InventoryKey Key_T;


  public:
    // ////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    const AirlineCode_T& getAirlineCode() const {
      return _key.getAirlineCode();
    }

    ForecastingMethod::EN_ForecastingMethod getForecastingMethod() const;

    UnconstrainingMethod::EN_UnconstrainingMethod getUnconstrainingMethod() const;

    PreOptimisationMethod::EN_PreOptimisationMethod getPreOptimisationMethod() const;

    OptimisationMethod::EN_OptimisationMethod getOptimisationMethod() const;

    PartnershipTechnique::EN_PartnershipTechnique getPartnershipTechnique() const;

    BomAbstract* const getParent() const {
      return _parent;
    }
```

```
00079
00081     const HolderMap_T& getHolderMap() const {
00082       return _holderMap;
00083     }
00084
00095     FlightDate* getFlightDate (const std::string& iFlightDateKeyStr) const;
00096
00107     FlightDate* getFlightDate (const FlightDateKey&) const;
00108
00112     AirlineFeature* getAirlineFeature () const {
00113       return _airlineFeature;
00114     }
00115
00116
00117   private:
00118     // /////////// Setters ////////////
00120     void setAirlineFeature (AirlineFeature& iAirlineFeature) {
00121       _airlineFeature = &iAirlineFeature;
00122     }
00123
00124
00125   public:
00126     // /////////// Display support methods /////////
00132     void toStream (std::ostream& ioOut) const {
00133       ioOut << toString();
00134     }
00135
00141     void fromStream (std::istream& ioIn) {
00142     }
00143
00147     std::string toString() const;
00148
00152     const std::string describeKey() const {
00153       return _key.toString();
00154     }
00155
00156
00157   public:
00158     // /////////// (Boost) Serialisation support methods /////////
00162     template<class Archive>
00163     void serialize (Archive& ar, const unsigned int iFileVersion);
00164
00165   private:
00173     void serialisationImplementationExport() const;
00174     void serialisationImplementationImport();
00175
00176
00177   protected:
00178     // /////////// Constructors and destructors /////////
00182     Inventory (const Key_T&);
00186     ~Inventory();
00187
00188   private:
00192     Inventory();
00196     Inventory (const Inventory&);
00197
00198
00199   protected:
00200     // /////////// Attributes /////////
00204     Key_T _key;
00205
00209     BomAbstract* _parent;
00210
00214     AirlineFeature* _airlineFeature;
00215
00219     HolderMap_T _holderMap;
00220   };
```

```
00221
00222 }
00223 #endif // __STDAIR_BOM_INVENTORY_HPP
00224
```

33.345 stdair/bom/InventoryKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/InventoryKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::InventoryKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::InventoryKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.346 stdair/bom/InventoryKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/InventoryKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  InventoryKey::InventoryKey() : _airlineCode (DEFAULT_AIRLINE_CODE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  InventoryKey::InventoryKey (const AirlineCode_T& iAirlineCode)
    : _airlineCode (iAirlineCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  InventoryKey::InventoryKey (const InventoryKey& iKey)
    : _airlineCode (iKey._airlineCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  InventoryKey::~InventoryKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void InventoryKey::toStream (std::ostream& ioOut) const {
    ioOut << "InventoryKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void InventoryKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string InventoryKey::toString() const {
    std::ostringstream oStr;
    oStr << _airlineCode;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void InventoryKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void InventoryKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void InventoryKey::serialize (Archive& ioArchive,
00069                                 const unsigned int iFileVersion) {
00070     ioArchive & _airlineCode;
00071   }
00072
00073   // ////////////////////////////////////////////////////////////////////
00074   // Explicit template instantiation
00075   namespace ba = boost::archive;
00076   template void InventoryKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00077                                                             unsigned int);
00078   template void InventoryKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00079                                                             unsigned int);
00080   // ////////////////////////////////////////////////////////////////////
00081
00082 }
```

33.347 stdair/bom/InventoryKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::InventoryKey

  Key of a given inventory, made of the airline code.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.348 stdair/bom/InventoryKey.hpp

```
00001 #ifndef __STDAIR_BOM_INVENTORYKEY_HPP
00002 #define __STDAIR_BOM_INVENTORYKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct InventoryKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     InventoryKey();
00035
00036   public:
00037     // /////////// Construction ///////////
00041     InventoryKey (const AirlineCode_T& iAirlineCode);
00042
00046     InventoryKey (const InventoryKey&);
00047
00051     ~InventoryKey();
00052
00053
00054     // /////////// Getters //////////
00058     const AirlineCode_T& getAirlineCode() const {
00059       return _airlineCode;
00060     }
00061
00062
00063   public:
00064     // /////////// Display support methods /////////
00070     void toStream (std::ostream& ioOut) const;
00071
00077     void fromStream (std::istream& ioIn);
00078
00088     const std::string toString() const;
00089
00090
00091   public:
00092     // /////////// (Boost) Serialisation support methods /////////
00096     template<class Archive>
00097     void serialize (Archive& ar, const unsigned int iFileVersion);
00098
00099   private:
00104     void serialisationImplementationExport() const;
00105     void serialisationImplementationImport();
00106
00107
00108   private:
00109     // ///////////////// Attributes ///////////////
00113     AirlineCode_T _airlineCode;
```

```
00114   };
00115
00116 }
00117 #endif // __STDAIR_BOM_INVENTORYKEY_HPP
```

33.349 stdair/bom/InventoryTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< Inventory * > stdair::InventoryList_T
- typedef std::map< const MapKey_T, Inventory * > stdair::InventoryMap_T

33.350 stdair/bom/InventoryTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_INVENTORYTYPES_HPP
00003 #define __STDAIR_BOM_INVENTORYTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // Stdair
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class Inventory;
00018
00020   typedef std::list<Inventory*> InventoryList_T;
00021
00023   typedef std::map<const MapKey_T, Inventory*> InventoryMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_INVENTORYTYPES_HPP
```

33.351 stdair/bom/key_types.hpp File Reference

```
#include <string>
#include <list>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::string stdair::MapKey_T
- typedef std::list< std::string > stdair::KeyList_T

33.352 stdair/bom/key_types.hpp

```
#ifndef __STDAIR_BOM_KEY_TYPES_HPP
#define __STDAIR_BOM_KEY_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <list>

namespace stdair {

  // ////////////////////// Type definitions ///////////
  typedef std::string MapKey_T;

  typedef std::list<std::string> KeyList_T;

}
#endif // __STDAIR_BOM_KEY_TYPES_HPP
```

33.353 stdair/bom/KeyAbstract.hpp File Reference

```
#include <iosfwd>
#include <string>
```

Classes

- struct stdair::KeyAbstract

  Base class for the keys of Business Object Model (BOM) layer.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template<class charT , class traits >
  std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::KeyAbstract &iKey)
- template<class charT , class traits >
  std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::KeyAbstract &ioKey)

33.353.1 Function Documentation

33.353.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::KeyAbstract & *iKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 74 of file KeyAbstract.hpp.

33.353.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::KeyAbstract & *ioKey*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 102 of file KeyAbstract.hpp.

References stdair::KeyAbstract::fromStream().

33.354 stdair/bom/KeyAbstract.hpp

```
00001
00007 #ifndef __STDAIR_BOM_KEYABSTRACT_HPP
00008 #define __STDAIR_BOM_KEYABSTRACT_HPP
00009
00010 // ////////////////////////////////////////////////////////////////////
00011 // Import section
00012 // ////////////////////////////////////////////////////////////////////
00013 // STL
00014 #include <iosfwd>
00015 #include <string>
00016
00017 namespace stdair {
00018
00027   struct KeyAbstract {
00028   public:
00029
00030     // /////////// Display support methods /////////
00036     virtual void toStream (std::ostream& ioOut) const {}
00037
00043     virtual void fromStream (std::istream& ioIn) {}
00044
00056     virtual const std::string toString() const { return std::string("Hello!"); }
00057
00061     virtual ~KeyAbstract() {}
00062   };
00063
00064 }
00065
00071 template <class charT, class traits>
00072 inline
00073 std::basic_ostream<charT, traits>&
00074 operator<< (std::basic_ostream<charT, traits>& ioOut,
00075             const stdair::KeyAbstract& iKey) {
00081   std::basic_ostringstream<charT,traits> ostr;
00082   ostr.copyfmt (ioOut);
00083   ostr.width (0);
00084
00085   // Fill string stream
00086   iKey.toStream (ostr);
00087
00088   // Print string stream
00089   ioOut << ostr.str();
00090
00091   return ioOut;
00092 }
00093
00099 template <class charT, class traits>
00100 inline
00101 std::basic_istream<charT, traits>&
00102 operator>> (std::basic_istream<charT, traits>& ioIn,
00103             stdair::KeyAbstract& ioKey) {
00104   // Fill Key object with input stream
00105   ioKey.fromStream (ioIn);
00106   return ioIn;
00107 }
00108
00109 #endif // __STDAIR_BOM_KEYABSTRACT_HPP
```

33.355 stdair/bom/LegCabin.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.356 stdair/bom/LegCabin.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>


namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  LegCabin::LegCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  LegCabin::LegCabin (const LegCabin& iLegCabin)
    : _key (iLegCabin._key), _parent (NULL),
      _offeredCapacity (iLegCabin._offeredCapacity),
      _physicalCapacity (iLegCabin._physicalCapacity),
      _soldSeat (iLegCabin._soldSeat),
      _committedSpace (iLegCabin._committedSpace),
      _availabilityPool (iLegCabin._availabilityPool),
      _availability (iLegCabin._availability),
      _currentBidPrice (iLegCabin._currentBidPrice),
      _dcsRegrade (iLegCabin._dcsRegrade),
      _au (iLegCabin._au),
      _upr (iLegCabin._upr),
      _nav (iLegCabin._nav),
      _gav (iLegCabin._gav),
      _acp (iLegCabin._acp),
      _etb (iLegCabin._etb),
      _staffNbOfBookings (iLegCabin._staffNbOfBookings),
      _wlNbOfBookings (iLegCabin._wlNbOfBookings),
      _groupNbOfBookings (iLegCabin._groupNbOfBookings) {
  }

  // ////////////////////////////////////////////////////////////////////
  LegCabin::LegCabin (const Key_T& iKey)
    : _key (iKey), _parent (NULL),
      _offeredCapacity (DEFAULT_CABIN_CAPACITY),
      _physicalCapacity (DEFAULT_CABIN_CAPACITY),
      _soldSeat (DEFAULT_CLASS_NB_OF_BOOKINGS),
      _committedSpace (DEFAULT_COMMITTED_SPACE),
      _availabilityPool (DEFAULT_AVAILABILITY),
      _availability (DEFAULT_AVAILABILITY),
      _currentBidPrice (DEFAULT_BID_PRICE),
      _bidPriceVector (DEFAULT_BID_PRICE_VECTOR),
      _dcsRegrade (DEFAULT_NULL_CAPACITY_ADJUSTMENT),
      _au (DEFAULT_CLASS_AUTHORIZATION_LEVEL),
      _upr (DEFAULT_NULL_UPR),
      _nav (DEFAULT_AVAILABILITY),
      _gav (DEFAULT_AVAILABILITY),
      _acp (DEFAULT_CLASS_OVERBOOKING_RATE),
      _etb (DEFAULT_NULL_BOOKING_NUMBER),
      _staffNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER),
      _wlNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER),
      _groupNbOfBookings (DEFAULT_NULL_BOOKING_NUMBER) {
```

```
}

// ////////////////////////////////////////////////////////////////////
LegCabin::~LegCabin() {
}

// ////////////////////////////////////////////////////////////////////
void LegCabin::setCapacities (const CabinCapacity_T& iCapacity) {
  _offeredCapacity = iCapacity;
  _physicalCapacity = iCapacity;
  setAvailabilityPool (iCapacity - _committedSpace);
}

// ////////////////////////////////////////////////////////////////////
const MapKey_T LegCabin::getFullerKey() const {
  const LegDate& lLegDate = BomManager::getParent<LegDate> (*this);

  const MapKey_T oFullKey =
    lLegDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER + getCabinCode();
  return oFullKey;
}

// ////////////////////////////////////////////////////////////////////
std::string LegCabin::toString() const {
  std::ostringstream oStr;
  oStr << describeKey();
  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const std::string LegCabin::displayVirtualClassList () const {
  std::ostringstream oStr;

  for (VirtualClassList_T::const_iterator itVC = _virtualClassList.begin();
       itVC != _virtualClassList.end(); ++itVC) {
    const VirtualClassStruct& lVC = *itVC;
    oStr << std::endl << "Yield: " << std::fixed << std::setprecision (2)
         << lVC.getYield()
         << ", Protection: " << std::fixed << std::setprecision (2)
         << lVC.getCumulatedProtection()
         << ", Booking limit: " << std::fixed << std::setprecision (2)
         << lVC.getCumulatedBookingLimit();
  }

  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
void LegCabin::updateFromReservation (const NbOfBookings_T& iNbOfBookings) {
  _committedSpace += iNbOfBookings;
  _availabilityPool = _offeredCapacity - _committedSpace;
}

// ////////////////////////////////////////////////////////////////////
void LegCabin::updateCurrentBidPrice() {
  const unsigned short lAvailabilityPool =
    static_cast<unsigned short> (std::floor (_availabilityPool));

  if (lAvailabilityPool >= 1) {
    const unsigned short lBidPriceVectorSize = _bidPriceVector.size();
    if (lBidPriceVectorSize >= lAvailabilityPool) {
      _currentBidPrice = _bidPriceVector.at (lAvailabilityPool - 1);
    }
  }
}

// ////////////////////////////////////////////////////////////////////
```

```
  void LegCabin::addDemandInformation (const YieldValue_T& iYield,
                                       const MeanValue_T& iMeanValue,
                                       const StdDevValue_T& iStdDevValue) {
    //
    const int lYieldLevel =
      static_cast<int> (std::floor (iYield + 0.5));

    //
    YieldLevelDemandMap_T::iterator itDemand =
      _yieldLevelDemandMap.find (lYieldLevel);

    if (itDemand == _yieldLevelDemandMap.end()) {
      MeanStdDevPair_T lMeanStdDevPair (iMeanValue,iStdDevValue);
      const bool hasInsertBeenSuccessful = _yieldLevelDemandMap.
        insert (YieldLevelDemandMap_T::value_type (lYieldLevel,
                                                   lMeanStdDevPair)).second;
      assert (hasInsertBeenSuccessful == true);

    } else {
      //
      MeanStdDevPair_T& lMeanStdDevPair = itDemand->second;
      MeanValue_T lMeanValue = iMeanValue + lMeanStdDevPair.first;
      StdDevValue_T lStdDevValue = iStdDevValue * iStdDevValue + lMeanStdDevPair.
    second * lMeanStdDevPair.second;
      lStdDevValue = std::sqrt (lStdDevValue);

      //
      lMeanStdDevPair = MeanStdDevPair_T (lMeanValue, lStdDevValue);
    }
  }

}
```

33.357 stdair/bom/LegCabin.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegCabinKey.hpp>
#include <stdair/bom/LegCabinTypes.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/VirtualClassTypes.hpp>
```

Classes

- class stdair::LegCabin

  Class representing the actual attributes for an airline leg-cabin.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.358 stdair/bom/LegCabin.hpp

```
#ifndef __STDAIR_BOM_LEGCABIN_HPP
#define __STDAIR_BOM_LEGCABIN_HPP
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegCabinKey.hpp>
#include <stdair/bom/LegCabinTypes.hpp>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/VirtualClassTypes.hpp>


namespace stdair {

  class LegCabin : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;

  public:
    // ////////// Type definitions ////////////
    typedef LegCabinKey Key_T;

  public:
    // /////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const CabinCode_T& getCabinCode() const {
      return _key.getCabinCode();
    }

    const MapKey_T getFullerKey() const;

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const CabinCapacity_T& getOfferedCapacity() const {
      return _offeredCapacity;
    }

    const CabinCapacity_T& getPhysicalCapacity() const {
      return _physicalCapacity;
    }

    const NbOfSeats_T& getSoldSeat() const {
      return _soldSeat;
    }

    const CommittedSpace_T& getCommittedSpace() const {
      return _committedSpace;
    }

```

```
00097     const Availability_T& getAvailabilityPool() const {
00098       return _availabilityPool;
00099     }
00100
00102     const Availability_T& getAvailability() const {
00103       return _availability;
00104     }
00105
00107     const BidPrice_T& getCurrentBidPrice() const {
00108       return _currentBidPrice;
00109     }
00110
00112     const BidPrice_T& getPreviousBidPrice() const {
00113       return _previousBidPrice;
00114     }
00115
00117     const BidPriceVector_T& getBidPriceVector() const {
00118       return _bidPriceVector;
00119     }
00120
00122     const CapacityAdjustment_T& getRegradeAdjustment() const {
00123       return _dcsRegrade;
00124     }
00125
00127     const AuthorizationLevel_T& getAuthorizationLevel() const {
00128       return _au;
00129     }
00130
00132     const UPR_T& getUPR() const {
00133       return _upr;
00134     }
00135
00137     const Availability_T& getNetAvailability() const {
00138       return _nav;
00139     }
00140
00142     const Availability_T& getGrossAvailability() const {
00143       return _gav;
00144     }
00145
00147     const OverbookingRate_T& getAvgCancellationPercentage() const {
00148       return _acp;
00149     }
00150
00152     const NbOfSeats_T& getETB() const {
00153       return _etb;
00154     }
00155
00157     const NbOfSeats_T& getStaffNbOfSeats() const {
00158       return _staffNbOfBookings;
00159     }
00160
00162     const NbOfSeats_T& getWLNbOfSeats() const {
00163       return _wlNbOfBookings;
00164     }
00165
00167     const NbOfSeats_T& getGroupNbOfSeats() const {
00168       return _groupNbOfBookings;
00169     }
00170
00172     VirtualClassList_T& getVirtualClassList() {
00173       return _virtualClassList;
00174     }
00175
00177     BidPriceVector_T& getBidPriceVector() {
00178       return _bidPriceVector;
00179     }
```

```
00180
00181
00183     const YieldLevelDemandMap_T& getYieldLevelDemandMap() {
00184       return _yieldLevelDemandMap;
00185     }
00186
00187
00188   public:
00189     // ///////////// Setters ///////////////
00191     void setCapacities (const CabinCapacity_T& iCapacity);
00192
00194     void setSoldSeat (const NbOfSeats_T& iSoldSeat) {
00195       _soldSeat = iSoldSeat;
00196     }
00197
00199     void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00200       _committedSpace = iCommittedSpace;
00201     }
00202
00204     void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00205       _availabilityPool = iAvailabilityPool;
00206     }
00207
00209     void setAvailability (const Availability_T& iAvailability) {
00210       _availability = iAvailability;
00211     }
00212
00214     void setCurrentBidPrice (const BidPrice_T& iBidPrice) {
00215       _currentBidPrice = iBidPrice;
00216     }
00217
00219     void setPreviousBidPrice (const BidPrice_T& iBidPrice) {
00220       _previousBidPrice = iBidPrice;
00221     }
00222
00224     void updatePreviousBidPrice () {
00225       _previousBidPrice = _currentBidPrice;
00226     }
00227
00229     void setRegradeAdjustment (const CapacityAdjustment_T& iRegradeAdjustment) {
00230       _dcsRegrade = iRegradeAdjustment;
00231     }
00232
00234     void setAuthorizationLevel (const AuthorizationLevel_T& iAU) {
00235       _au = iAU;
00236     }
00237
00239     void setUPR (const UPR_T& iUPR) {
00240       _upr = iUPR;
00241     }
00242
00244     void setNetAvailability (const Availability_T& iNAV) {
00245       _nav = iNAV;
00246     }
00247
00249     void setGrossAvailability (const Availability_T& iGAV) {
00250       _gav = iGAV;
00251     }
00252
00254     void setAvgCancellationPercentage (const OverbookingRate_T& iACP) {
00255       _acp = iACP;
00256     }
00257
00259     void setETB (const NbOfSeats_T& iETB) {
00260       _etb = iETB;
00261     }
00262
```

```
00264     void setStaffNbOfSeats (const NbOfSeats_T& iStaffSeats) {
00265       _staffNbOfBookings = iStaffSeats;
00266     }
00267
00269     void setWLNbOfSeats (const NbOfSeats_T& iWLSeats) {
00270       _wlNbOfBookings = iWLSeats;
00271     }
00272
00274     void setGroupNbOfSeats (const NbOfSeats_T& iGroupSeats) {
00275       _groupNbOfBookings = iGroupSeats;
00276     }
00277
00279     void updateCurrentBidPrice();
00280
00281
00282   public:
00283     // /////////// Display support methods /////////
00288     void toStream (std::ostream& ioOut) const {
00289       ioOut << toString();
00290     }
00291
00296     void fromStream (std::istream& ioIn) {
00297     }
00298
00302     std::string toString() const;
00303
00307     const std::string describeKey() const {
00308       return _key.toString();
00309     }
00310
00314     const std::string displayVirtualClassList() const;
00315
00316
00317   public:
00318     // /////////// Business methods //////////
00322     void updateFromReservation (const NbOfBookings_T&);
00323
00327     void addVirtualClass (const VirtualClassStruct& iVC) {
00328       _virtualClassList.push_back (iVC);
00329     }
00330
00334     void emptyVirtualClassList() {
00335       _virtualClassList.clear();
00336     }
00337
00341     void emptyBidPriceVector() {
00342       _bidPriceVector.clear();
00343     }
00344
00348     void addDemandInformation (const YieldValue_T&, const MeanValue_T&,
00349                                const StdDevValue_T&);
00350
00354     void emptyYieldLevelDemandMap() {
00355       _yieldLevelDemandMap.clear();
00356     }
00357
00358
00359   protected:
00360     // ////////// Constructors and destructors /////////
00364     LegCabin (const Key_T&);
00368     ~LegCabin();
00369
00370
00371   private:
00375     LegCabin();
00379     LegCabin (const LegCabin&);
00380
```

```
00381
00382   protected:
00383     // ////////// Attributes ////////
00387     Key_T _key;
00388
00392     BomAbstract* _parent;
00393
00397     HolderMap_T _holderMap;
00398
00400     CabinCapacity_T _offeredCapacity;
00401
00403     CabinCapacity_T _physicalCapacity;
00404
00406     NbOfSeats_T  _soldSeat;
00407
00408     /* Committed space. */
00409     CommittedSpace_T  _committedSpace;
00410
00412     Availability_T _availabilityPool;
00413
00415     Availability_T _availability;
00416
00418     BidPrice_T _currentBidPrice;
00419
00421     BidPrice_T _previousBidPrice;
00422
00424     BidPriceVector_T _bidPriceVector;
00425
00427     VirtualClassList_T _virtualClassList;
00428
00430     YieldLevelDemandMap_T _yieldLevelDemandMap;
00431
00432
00433   public:
00435     CapacityAdjustment_T _dcsRegrade;
00436
00438     AuthorizationLevel_T _au;
00439
00441     UPR_T _upr;
00442
00444     Availability_T _nav;
00445
00447     Availability_T _gav;
00448
00450     OverbookingRate_T _acp;
00451
00453     NbOfSeats_T _etb;
00454
00456     NbOfSeats_T _staffNbOfBookings;
00457
00459     NbOfSeats_T _wlNbOfBookings;
00460
00462     NbOfSeats_T _groupNbOfBookings;
00463   };
00464
00465 }
00466 #endif // __STDAIR_BOM_LEGCABIN_HPP
00467
```

33.359 stdair/bom/LegCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegCabinKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.360 stdair/bom/LegCabinKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/LegCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   LegCabinKey::LegCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   LegCabinKey::LegCabinKey (const CabinCode_T& iCabinCode)
00024     : _cabinCode (iCabinCode) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   LegCabinKey::LegCabinKey (const LegCabinKey& iKey)
00029     : _cabinCode (iKey._cabinCode) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   LegCabinKey::~LegCabinKey () {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void LegCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "LegCabinKey: " << toString() << std::endl;
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void LegCabinKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string LegCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void LegCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void LegCabinKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void LegCabinKey::serialize (Archive& ioArchive,
00069                                const unsigned int iFileVersion) {
00074     ioArchive & _cabinCode;
00075   }
00076
00077 }
```

33.361 stdair/bom/LegCabinKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::LegCabinKey

  Key of a given leg-cabin, made of a cabin code (only).

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.362 stdair/bom/LegCabinKey.hpp

```
00001 #ifndef __STDAIR_BOM_LEGCABINKEY_HPP
00002 #define __STDAIR_BOM_LEGCABINKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct LegCabinKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     LegCabinKey();
00035
00036   public:
00040     LegCabinKey (const CabinCode_T& iCabinCode);
00041
00045     LegCabinKey (const LegCabinKey&);
00046
00050     ~LegCabinKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const CabinCode_T& getCabinCode() const {
00057       return _cabinCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     CabinCode_T _cabinCode;
```

```
  };

}
#endif // __STDAIR_BOM_LEGCABINKEY_HPP
```

33.363 stdair/bom/LegCabinTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< LegCabin * > stdair::LegCabinList_T
- typedef std::map< const MapKey_T, LegCabin * > stdair::LegCabinMap_T

33.364 stdair/bom/LegCabinTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_LEGCABINTYPES_HPP
#define __STDAIR_BOM_LEGCABINTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class LegCabin;

  typedef std::list<LegCabin*> LegCabinList_T;

  typedef std::map<const MapKey_T, LegCabin*> LegCabinMap_T;

}
#endif // __STDAIR_BOM_LEGCABINTYPES_HPP

```

33.365 stdair/bom/LegDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.366 stdair/bom/LegDate.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/LegDate.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  LegDate::LegDate() : _key (DEFAULT_ORIGIN), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  LegDate::LegDate (const LegDate& iLegDate) :
    _key (iLegDate._key),
    _parent (NULL),
    _offPoint (iLegDate._offPoint),
    _boardingDate (iLegDate._boardingDate),
    _boardingTime (iLegDate._boardingTime),
    _offDate (iLegDate._offDate),
    _offTime (iLegDate._offTime ),
    _elapsedTime (iLegDate._elapsedTime),
    _distance (iLegDate._distance),
    _capacity (iLegDate._capacity) {
  }

  // ////////////////////////////////////////////////////////////////////
  LegDate::LegDate (const Key_T& iKey)
    : _key (iKey), _parent (NULL), _distance (DEFAULT_DISTANCE_VALUE),
      _capacity (DEFAULT_CABIN_CAPACITY) {
  }

  // ////////////////////////////////////////////////////////////////////
  LegDate::~LegDate () {
  }

  // ////////////////////////////////////////////////////////////////////
  const AirlineCode_T& LegDate::getAirlineCode() const {
    const FlightDate* lFlightDate_ptr =
      static_cast<const FlightDate*> (getParent());
    assert (lFlightDate_ptr != NULL);
    return lFlightDate_ptr->getAirlineCode();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string LegDate::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string LegDate::describeRoutingKey() const {
    const FlightDate* lFlightDate_ptr =
      static_cast<const FlightDate*> (getParent());
```

```
    assert (lFlightDate_ptr != NULL);
    std::ostringstream oStr;
    oStr << _operatingAirlineCode << DEFAULT_KEY_FLD_DELIMITER
         << _operatingFlightNumber << DEFAULT_KEY_FLD_DELIMITER
         << lFlightDate_ptr->getDepartureDate() << DEFAULT_KEY_FLD_DELIMITER
         << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  LegCabin* LegDate::getLegCabin (const std::string& iLegCabinKeyStr) const {
    LegCabin* oLegCabin_ptr =
      BomManager::getObjectPtr<LegCabin> (*this, iLegCabinKeyStr);
    return oLegCabin_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  LegCabin* LegDate::getLegCabin (const LegCabinKey& iLegCabinKey) const {
    return getLegCabin (iLegCabinKey.toString());
  }

  // ////////////////////////////////////////////////////////////////////
  const Duration_T LegDate::getTimeOffset() const {
    // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
    //              - ElapsedTime
    Duration_T oTimeOffset = (_offTime - _boardingTime);

    const DateOffset_T& lDateOffset = getDateOffset();

    const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);

    oTimeOffset += lDateOffsetInHours - _elapsedTime;

    return oTimeOffset;
  }

  // ////////////////////////////////////////////////////////////////////
  void LegDate::setElapsedTime (const Duration_T& iElapsedTime) {
    // Set Elapsed time
    _elapsedTime = iElapsedTime;

    // Update distance according to the mean plane speed
    updateDistanceFromElapsedTime();
  }

  // ////////////////////////////////////////////////////////////////////
  void LegDate::updateDistanceFromElapsedTime() {
    //
    const double lElapseInHours =
      static_cast<const double> (_elapsedTime.hours());

    // Normally, Distance_T is an unsigned long int
    const Distance_T lDistance =
      static_cast<const Distance_T> (DEFAULT_FLIGHT_SPEED * lElapseInHours);

    _distance = lDistance;
  }

}
```

33.367 stdair/bom/LegDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/LegDateTypes.hpp>
```

Classes

- class stdair::LegDate

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.368 stdair/bom/LegDate.hpp

```
#ifndef __STDAIR_BOM_LEGDATE_HPP
#define __STDAIR_BOM_LEGDATE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/LegDateTypes.hpp>

namespace stdair {

  struct LegCabinKey;
  class LegCabin;

  class LegDate : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;

  public:
    // ////////// Type definitions ////////////
    typedef LegDateKey Key_T;


  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const AirportCode_T& getBoardingPoint() const {
      return _key.getBoardingPoint();
    }

    const AirlineCode_T& getAirlineCode() const;

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    LegCabin* getLegCabin (const std::string& iLegCabinKeyStr) const;

    LegCabin* getLegCabin (const LegCabinKey&) const;

    const AirportCode_T& getOffPoint() const {
      return _offPoint;
    }

    const Date_T& getBoardingDate() const {
      return _boardingDate;
    }

    const Duration_T& getBoardingTime() const {
      return _boardingTime;
    }
```

```
00107
00109     const Date_T& getOffDate() const {
00110       return _offDate;
00111     }
00112
00114     const Duration_T& getOffTime() const {
00115       return _offTime;
00116     }
00117
00119     const Duration_T& getElapsedTime() const {
00120       return _elapsedTime;
00121     }
00122
00124     const Distance_T& getDistance() const {
00125       return _distance;
00126     }
00127
00129     const CabinCapacity_T& getCapacity() const {
00130       return _capacity;
00131     }
00132
00134     const DateOffset_T getDateOffset() const {
00135       return _offDate - _boardingDate;
00136     }
00137
00142     const Duration_T getTimeOffset() const;
00143
00144
00145   public:
00146     // ///////// Setters //////////
00148     void setOffPoint (const AirportCode_T& iOffPoint) {
00149       _offPoint = iOffPoint;
00150     }
00151
00153     void setBoardingDate (const Date_T& iBoardingDate) {
00154       _boardingDate = iBoardingDate;
00155     }
00156
00158     void setBoardingTime (const Duration_T& iBoardingTime) {
00159       _boardingTime = iBoardingTime;
00160     }
00161
00163     void setOffDate (const Date_T& iOffDate) {
00164       _offDate = iOffDate;
00165     }
00166
00168     void setOffTime (const Duration_T& iOffTime) {
00169       _offTime = iOffTime;
00170     }
00171
00173     void setElapsedTime (const Duration_T&);
00174
00176     void setOperatingAirlineCode (const AirlineCode_T& iAirlineCode) {
00177       _operatingAirlineCode = iAirlineCode;
00178     }
00179
00181     void setOperatingFlightNumber (const FlightNumber_T& iFlightNumber) {
00182       _operatingFlightNumber = iFlightNumber;
00183     }
00184
00185   private:
00187     void updateDistanceFromElapsedTime();
00188
00189
00190   public:
00191     // /////////// Display support methods /////////
00194     void toStream (std::ostream& ioOut) const {
```

```
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }

    const std::string describeRoutingKey() const;

  protected:
    // ////////// Constructors and destructors /////////
    LegDate (const Key_T&);
    virtual ~LegDate();

  private:
    LegDate();
    LegDate (const LegDate&);


  protected:
    // ////////// Attributes /////////
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;

    AirportCode_T _offPoint;

    Date_T _boardingDate;

    Duration_T _boardingTime;

    Date_T _offDate;

    Duration_T _offTime;

    Duration_T _elapsedTime;

    Distance_T _distance;

    CabinCapacity_T _capacity;

    AirlineCode_T _operatingAirlineCode;

    FlightNumber_T _operatingFlightNumber;
  };

}
#endif // __STDAIR_BOM_LEGDATE_HPP
```

33.369 stdair/bom/LegDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegDateKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.370 stdair/bom/LegDateKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/LegDateKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  LegDateKey::LegDateKey() : _boardingPoint (DEFAULT_ORIGIN) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  LegDateKey::LegDateKey (const AirportCode_T& iBoardingPoint)
    : _boardingPoint (iBoardingPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  LegDateKey::LegDateKey (const LegDateKey& iKey)
    : _boardingPoint (iKey._boardingPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  LegDateKey::~LegDateKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void LegDateKey::toStream (std::ostream& ioOut) const {
    ioOut << "LegDateKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void LegDateKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string LegDateKey::toString() const {
    std::ostringstream oStr;
    oStr << _boardingPoint;
    return oStr.str();
  }

}
```

33.371 stdair/bom/LegDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::LegDateKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.372 stdair/bom/LegDateKey.hpp

```
#ifndef __STDAIR_BOM_LEGDATEKEY_HPP
#define __STDAIR_BOM_LEGDATEKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace stdair {

  struct LegDateKey : public KeyAbstract {

    // /////////// Constructors and destructors ///////////
  private:
    LegDateKey();

  public:
    LegDateKey (const AirportCode_T& iBoardingPoint);
    LegDateKey (const LegDateKey&);
    ~LegDateKey();


    // /////////// Getters //////////
    const AirportCode_T& getBoardingPoint() const {
      return _boardingPoint;
    }


    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;


  private:
    // ///////////////// Attributes ///////////////
    AirportCode_T _boardingPoint;
  };

}
#endif // __STDAIR_BOM_LEGDATEKEY_HPP
```

33.373 stdair/bom/LegDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< LegDate * > stdair::LegDateList_T
- typedef std::map< const MapKey_T, LegDate * > stdair::LegDateMap_T

33.374 stdair/bom/LegDateTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_LEGDATETYPES_HPP
00003 #define __STDAIR_BOM_LEGDATETYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class LegDate;
00018
00020   typedef std::list<LegDate*> LegDateList_T;
00021
00023   typedef std::map<const MapKey_T, LegDate*> LegDateMap_T;
00024 }
00025 #endif // __STDAIR_BOM_LEGDATETYPES_HPP
00026
```

33.375 stdair/bom/NestingNode.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/NestingNode.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.376 stdair/bom/NestingNode.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
//STDAIR
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/NestingNode.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  NestingNode::NestingNode () :
    _key (DEFAULT_NESTING_NODE_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  NestingNode::NestingNode (const NestingNode& iNestingNode)
  : _key (DEFAULT_NESTING_NODE_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  NestingNode::NestingNode (const Key_T& iKey) : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  NestingNode::~NestingNode() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string NestingNode::toString () const {
    std::ostringstream oStr;
    oStr << describeKey();

    oStr << _yield << std::endl;

    return oStr.str();
  }

}
```

33.377 stdair/bom/NestingNode.hpp File Reference

```
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/NestingNodeKey.hpp>
```

Classes

- class stdair::NestingNode

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.378 stdair/bom/NestingNode.hpp

```
00001 #ifndef __STDAIR_BOM_NESTINGNODE_HPP
00002 #define __STDAIR_BOM_NESTINGNODE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <cmath>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/NestingNodeKey.hpp>
00015
00017 namespace boost {
00018   namespace serialization {
00019     class access;
00020   }
00021 }
00022
00023 namespace stdair {
00024
00029   class NestingNode : public BomAbstract {
00030     template <typename BOM> friend class FacBom;
00031     friend class FacBomManager;
00032     friend class boost::serialization::access;
00033
00034   public:
00035     // ////////// Type definitions ////////////
00039     typedef NestingNodeKey Key_T;
00040
00041   public:
00042     // ////////////////// Getters ////////////////////////
00044     const Key_T& getKey() const {
00045       return _key;
00046     }
00047
00049     BomAbstract* const getParent() const {
00050       return _parent;
00051     }
00052
00056     const HolderMap_T& getHolderMap() const {
00057       return _holderMap;
00058     }
00059
00061     const Yield_T& getYield() const {
00062       return _yield;
00063     }
00064
00065   public:
00066     // ////////////////////// Setters ////////////////////
00068     void setYield (const Yield_T& iYield) {
00069       _yield = iYield;
00070     }
00071
00072
00073   public:
00074     // /////////// Display support methods /////////
00080     void toStream (std::ostream& ioOut) const {
00081       ioOut << toString();
00082     }
00083
00089     void fromStream (std::istream& ioIn) {
00090     }
```

```
00091
00095     std::string toString() const;
00096
00100     const std::string describeKey() const {
00101       return _key.toString();
00102     }
00103
00104
00105   public:
00106     // /////////// (Boost) Serialisation support methods /////////
00110     template<class Archive>
00111     void serialize (Archive& ar, const unsigned int iFileVersion);
00112
00113   private:
00121     void serialisationImplementationExport() const;
00122     void serialisationImplementationImport();
00123
00124
00125   protected:
00126     // /////////// Constructors and destructor. ////////////
00130     NestingNode (const Key_T&);
00131
00135     virtual ~NestingNode();
00136
00137   private:
00141     NestingNode();
00142
00146     NestingNode (const NestingNode&);
00147
00148
00149   private:
00150     // ///////////// Attributes ////////////
00154     Key_T _key;
00155
00159     BomAbstract* _parent;
00160
00164     HolderMap_T _holderMap;
00165
00169     Yield_T _yield;
00170
00171   };
00172 }
00173 #endif // __STDAIR_BOM_NESTINGNODE_HPP
```

33.379 stdair/bom/NestingNodeKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/NestingNodeKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::NestingNodeKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::NestingNodeKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.380 stdair/bom/NestingNodeKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/NestingNodeKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   NestingNodeKey::NestingNodeKey() : _nestingNodeCode (DEFAULT_POLICY_CODE) {
00019     assert (false);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   NestingNodeKey::NestingNodeKey (const NestingNodeKey& iNestingNodeKey)
00024     : _nestingNodeCode (iNestingNodeKey._nestingNodeCode) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   NestingNodeKey::NestingNodeKey (const NestingNodeCode_T& iNestingNodeCode)
00029     : _nestingNodeCode (iNestingNodeCode) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   NestingNodeKey::~NestingNodeKey() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void NestingNodeKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "NestingNodeKey: " << toString();
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void NestingNodeKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string NestingNodeKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _nestingNodeCode;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void NestingNodeKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void NestingNodeKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void NestingNodeKey::serialize (Archive& ioArchive,
00069                                   const unsigned int iFileVersion) {
00074     ioArchive & _nestingNodeCode;
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   // Explicit template instantiation
00079   namespace ba = boost::archive;
00080   template void NestingNodeKey::serialize<ba::text_oarchive> (ba::text_oarchive&,

00081                                                              unsigned int);
00082   template void NestingNodeKey::serialize<ba::text_iarchive> (ba::text_iarchive&,

00083                                                              unsigned int);
00084   // ////////////////////////////////////////////////////////////////////
00085
00086 }
```

33.381 stdair/bom/NestingNodeKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::NestingNodeKey

  Key of a given policy, made of a policy code.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.382 stdair/bom/NestingNodeKey.hpp

```
00001 #ifndef __STDAIR_BOM_NESTINGNODEKEY_HPP
00002 #define __STDAIR_BOM_NESTINGNODEKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct NestingNodeKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     NestingNodeKey();
00035
00036   public:
00040     NestingNodeKey (const NestingNodeCode_T& iNestingNodeCode);
00041
00045     NestingNodeKey (const NestingNodeKey&);
00046
00050     ~NestingNodeKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const NestingNodeCode_T& getNestingNodeCode () const {
00057       return _nestingNodeCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     NestingNodeCode_T _nestingNodeCode;
```

```
  };

}
#endif // __STDAIR_BOM_NESTINGNODEKEY_HPP
```

33.383 stdair/bom/NestingNodeTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< NestingNode * > stdair::NestingNodeList_T
- typedef std::map< const MapKey_T, NestingNode * > stdair::NestingNodeMap_T

33.384 stdair/bom/NestingNodeTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_NESTINGNODETYPES_HPP
#define __STDAIR_BOM_NESTINGNODETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class NestingNode;

  typedef std::list<NestingNode*> NestingNodeList_T;

  typedef std::map<const MapKey_T, NestingNode*> NestingNodeMap_T;

}
#endif // __STDAIR_BOM_NESTINGNODETYPES_HPP
```

33.385 stdair/bom/NestingStructureKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/NestingStructureKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::NestingStructureKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::NestingStructureKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.386 stdair/bom/NestingStructureKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/NestingStructureKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  NestingStructureKey::NestingStructureKey() : _nestingStructureCode (DEFAULT_NESTING_STRUCTURE_CODE) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  NestingStructureKey::NestingStructureKey (const NestingStructureKey& iNestingStructureKey)
    : _nestingStructureCode (iNestingStructureKey._nestingStructureCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  NestingStructureKey::NestingStructureKey (const NestingStructureCode_T& iNestingStructureCode)
    : _nestingStructureCode (iNestingStructureCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  NestingStructureKey::~NestingStructureKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void NestingStructureKey::toStream (std::ostream& ioOut) const {
    ioOut << "NestingStructureKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void NestingStructureKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string NestingStructureKey::toString() const {
    std::ostringstream oStr;
    oStr << _nestingStructureCode;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void NestingStructureKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void NestingStructureKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
```

```
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void NestingStructureKey::serialize (Archive& ioArchive,
                                       const unsigned int iFileVersion) {
    ioArchive & _nestingStructureCode;
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template void NestingStructureKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                                    unsigned int);
  template void NestingStructureKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                                    unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

33.387 stdair/bom/NestingStructureKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::NestingStructureKey

  Key of a given policy, made of a policy code.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.388 stdair/bom/NestingStructureKey.hpp

```
00001 #ifndef __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP
00002 #define __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct NestingStructureKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     NestingStructureKey();
00035
00036   public:
00040     NestingStructureKey (const NestingStructureCode_T& iNestingStructureCode);
00041
00045     NestingStructureKey (const NestingStructureKey&);
00046
00050     ~NestingStructureKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const NestingStructureCode_T& getNestingStructureCode () const {
00057       return _nestingStructureCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     NestingStructureCode_T _nestingStructureCode;
```

```
  };

}
#endif // __STDAIR_BOM_NESTINGSTRUCTUREKEY_HPP
```

33.389 stdair/bom/OnDDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/OnDDate.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.390 stdair/bom/OnDDate.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/OnDDate.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  OnDDate::OnDDate()
    : _key (DEFAULT_OND_STRING_LIST), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDate::OnDDate (const OnDDate& iOnDDate)
    : _key (iOnDDate.getKey()), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDate::OnDDate (const Key_T& iKey)
  : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDate::~OnDDate() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string OnDDate::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const AirlineCode_T& OnDDate::getAirlineCode() const {
    const Inventory* lInventory_ptr =
      static_cast<const Inventory*> (getParent());
    assert (lInventory_ptr != NULL);
    return lInventory_ptr->getAirlineCode();
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDate::
  setDemandInformation (const CabinClassPairList_T& iCabinClassPairList,
                        const YieldDemandPair_T& iYieldDemandPair) {
    std::ostringstream oStr;
    for(CabinClassPairList_T::const_iterator itCCP = iCabinClassPairList.begin();

        itCCP != iCabinClassPairList.end(); ++itCCP) {
      oStr << itCCP->first << ":" << itCCP->second << ";";
    }
    std::string lCabinClassPath = oStr.str();
    StringDemandStructMap_T::iterator it =
      _classPathDemandMap.find(lCabinClassPath);
    if (it == _classPathDemandMap.end()) {
      const StringDemandStructPair_T lPairStringDemandChar (lCabinClassPath,
```

```
                                                                iYieldDemandPair);
      _classPathDemandMap.insert (lPairStringDemandChar);
      const StringCabinClassPair_T lStringCabinClassPair (lCabinClassPath,
                                                          iCabinClassPairList);
      _stringCabinClassPairListMap.insert (lStringCabinClassPair);
    } else {
        it->second = iYieldDemandPair;
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDate::setTotalForecast (const CabinCode_T& iCabinCode,
                                  const WTPDemandPair_T& iWTPDemandPair) {

    CabinForecastMap_T::iterator it =
      _cabinForecastMap.find (iCabinCode);
    if (it == _cabinForecastMap.end()) {
      const CabinForecastPair_T lPairCabinForecastChar (iCabinCode,
                                                        iWTPDemandPair);
      _cabinForecastMap.insert (lPairCabinForecastChar);
    } else {
      assert (false);
    }
  }

}
```

33.391 stdair/bom/OnDDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/OnDDateTypes.hpp>
```

Classes

- class stdair::OnDDate

  Class representing the actual attributes for an airline flight-date.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.392 stdair/bom/OnDDate.hpp

```
#ifndef __STDAIR_BOM_ONDDATE_HPP
#define __STDAIR_BOM_ONDDATE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/OnDDateTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  class OnDDate : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef OnDDateKey Key_T;


  public:
    // /////////// Getters ///////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const AirlineCode_T& getAirlineCode() const;


    const stdair::Date_T getDate() const {
      return _key.getDate();
    }

    const stdair::AirportCode_T getOrigin() const {
      return _key.getOrigin();
    }

    const stdair::AirportCode_T getDestination() const {
      return _key.getDestination();
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
```

```
    }

    const StringDemandStructMap_T& getDemandInfoMap () const {
      return _classPathDemandMap;
    }

    const CabinForecastMap_T& getTotalForecastMap () const {
      return _cabinForecastMap;
    }

    const WTPDemandPair_T& getTotalForecast (const CabinCode_T& iCC) const {
      assert (_cabinForecastMap.find(iCC)!=_cabinForecastMap.end());
      return _cabinForecastMap.find(iCC)->second;
    }

    const CabinClassPairList_T& getCabinClassPairList (const std::string& iStr) c
    onst {
      assert (_stringCabinClassPairListMap.find(iStr)!=
    _stringCabinClassPairListMap.end());
      return _stringCabinClassPairListMap.find(iStr)->second;
    }

    const short getNbOfSegments () const {
      return _key.getNbOfSegments();
    }

  public:
    // /////////// Setters ///////////////
    void setDemandInformation (const CabinClassPairList_T&,
                               const YieldDemandPair_T&);


    void setTotalForecast (const CabinCode_T&,
                           const WTPDemandPair_T&);


  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementation();


  protected:
    // ////////// Constructors and destructors /////////
    OnDDate (const Key_T&);

    virtual ~OnDDate();

```

```
00200   private:
00204     OnDDate();
00205
00209     OnDDate (const OnDDate&);
00210
00211
00212   protected:
00213     // ////////// Attributes ////////
00217     Key_T _key;
00218
00222     BomAbstract* _parent;
00223
00227     HolderMap_T _holderMap;
00228
00232     StringDemandStructMap_T _classPathDemandMap;
00233
00237     StringCabinClassPairListMap_T _stringCabinClassPairListMap;
00238
00242     CabinForecastMap_T _cabinForecastMap;
00243   };
00244
00245 }
00246 #endif // __STDAIR_BOM_ONDDATE_HPP
```

33.393 stdair/bom/OnDDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/date_time/gregorian/formatters.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/BomDisplay.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::OnDDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::OnDDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.394 stdair/bom/OnDDateKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost Date-Time
#include <boost/date_time/gregorian/formatters.hpp>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/OnDDateKey.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/BomDisplay.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  OnDDateKey::OnDDateKey()
    : _OnDStringList (DEFAULT_OND_STRING_LIST) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDateKey::OnDDateKey (const OnDStringList_T& iOnDStringList)
    : _OnDStringList (iOnDStringList) {
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDateKey::OnDDateKey (const OnDDateKey& iKey)
    : _OnDStringList (iKey._OnDStringList) {
  }

  // ////////////////////////////////////////////////////////////////////
  OnDDateKey::~OnDDateKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  const Date_T OnDDateKey::getDate() const {
    assert(_OnDStringList.empty() == false);
    const OnDString_T& lFrontOnDString = _OnDStringList.front();
    return BomKeyManager::extractFlightDateKey (lFrontOnDString).
getDepartureDate();
  }

  // ////////////////////////////////////////////////////////////////////
  const AirportCode_T OnDDateKey::getOrigin() const {
    assert(_OnDStringList.empty() == false);
    const OnDString_T& lFrontOnDString = _OnDStringList.front();
    return BomKeyManager::extractSegmentDateKey (lFrontOnDString).
getBoardingPoint();
  }

  // ////////////////////////////////////////////////////////////////////
  const AirportCode_T OnDDateKey::getDestination() const {
    assert(_OnDStringList.empty() == false);
    const OnDString_T& lLastOnDString = _OnDStringList.back();
```

```
    return BomKeyManager::extractSegmentDateKey (lLastOnDString).getOffPoint();
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDateKey::toStream (std::ostream& ioOut) const {
    ioOut << "OnDDateKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDateKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string OnDDateKey::toString() const {
    std::ostringstream oStr;
    for (OnDStringList_T::const_iterator itOnDString = _OnDStringList.begin();
         itOnDString != _OnDStringList.end(); ++itOnDString){
      oStr << *itOnDString << " ";
    }
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDateKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void OnDDateKey::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void OnDDateKey::serialize (Archive& ioArchive,
                              const unsigned int iFileVersion) {
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template void OnDDateKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                          unsigned int);
  template void OnDDateKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                          unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

33.395 stdair/bom/OnDDateKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::OnDDateKey

  Key of a given O&D-date, made of a list of OnD strings. a OnD string contains the airline code, the flight number, the date and the segment (origin and destination).

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.396 stdair/bom/OnDDateKey.hpp

```
00001 #ifndef __STDAIR_BOM_ONDDATEKEY_HPP
00002 #define __STDAIR_BOM_ONDDATEKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/stdair_date_time_types.hpp>
00014 #include <stdair/bom/KeyAbstract.hpp>
00015
00016 namespace stdair {
00017
00023   struct OnDDateKey : public KeyAbstract {
00024     friend class boost::serialization::access;
00025
00026     // /////////// Constructors and destructors ///////////
00027   private:
00031     OnDDateKey();
00032
00033   public:
00037     OnDDateKey (const OnDStringList_T&);
00038
00042     OnDDateKey (const OnDDateKey&);
00043
00047     ~OnDDateKey();
00048
00049
00050   public:
00051     // /////////// Getters //////////
00055     const Date_T getDate() const;
00056
00060     const AirportCode_T getOrigin() const;
00061
00065     const AirportCode_T getDestination() const;
00066
00070     const short getNbOfSegments () const {
00071       return _OnDStringList.size();
00072     }
00073
00074   public:
00075     // /////////// Display support methods /////////
00081     void toStream (std::ostream& ioOut) const;
00082
00088     void fromStream (std::istream& ioIn);
00089
00099     const std::string toString() const;
00100
00101
00102   public:
00103     // /////////// (Boost) Serialisation support methods /////////
00107     template<class Archive>
00108     void serialize (Archive& ar, const unsigned int iFileVersion);
00109
00110   private:
00115     void serialisationImplementationExport() const;
00116     void serialisationImplementationImport();
00117
00118
00119   private:
00120     // ///////////////// Attributes ///////////////
```

```
    OnDStringList_T _OnDStringList;

  };

}
#endif // __STDAIR_BOM_ONDDATEKEY_HPP
```

33.397 stdair/bom/OnDDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_demand_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< OnDDate * > stdair::OnDDateList_T
- typedef std::map< const MapKey_T, OnDDate * > stdair::OnDDateMap_T
- typedef std::pair< std::string, YieldDemandPair_T > stdair::StringDemandStructPair_T
- typedef std::map< std::string, YieldDemandPair_T > stdair::StringDemandStructMap_T
- typedef std::map< std::string, CabinClassPairList_T > stdair::StringCabinClassPairListMap_T
- typedef std::pair< std::string, CabinClassPairList_T > stdair::StringCabinClassPair_T
- typedef std::map< CabinCode_T, WTPDemandPair_T > stdair::CabinForecastMap_T
- typedef std::pair< CabinCode_T, WTPDemandPair_T > stdair::CabinForecastPair_T

33.398 stdair/bom/OnDDateTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_ONDDATETYPES_HPP
#define __STDAIR_BOM_ONDDATETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// Stdair
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_demand_types.hpp>

namespace stdair {

  // Forward declarations.
  class OnDDate;

  typedef std::list<OnDDate*> OnDDateList_T;

  typedef std::map<const MapKey_T, OnDDate*> OnDDateMap_T;

  typedef std::pair<std::string, YieldDemandPair_T> StringDemandStructPair_T;
  typedef std::map<std::string, YieldDemandPair_T> StringDemandStructMap_T;

  typedef std::map<std::string, CabinClassPairList_T>
StringCabinClassPairListMap_T;
  typedef std::pair<std::string, CabinClassPairList_T> StringCabinClassPair_T;

  typedef std::map<CabinCode_T, WTPDemandPair_T> CabinForecastMap_T;
  typedef std::pair<CabinCode_T, WTPDemandPair_T> CabinForecastPair_T;

}
#endif // __STDAIR_BOM_ONDDATETYPES_HPP
```

33.399 stdair/bom/OptimisationNotificationStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/OptimisationNotificationStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.400 stdair/bom/OptimisationNotificationStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/OptimisationNotificationStruct.hpp>
00009
00010 namespace stdair {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   OptimisationNotificationStruct::OptimisationNotificationStruct()
00014     :  _partySize (0), _stayDuration (0), _wtp (0.0), _valueOfTime (0.0) {
00015     assert (false);
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   OptimisationNotificationStruct::
00020   OptimisationNotificationStruct (const OptimisationNotificationStruct& iOptimisa
      tionNotification)
00021     : _origin (iOptimisationNotification._origin),
00022       _destination (iOptimisationNotification._destination),
00023       _pos (iOptimisationNotification._pos),
00024       _preferredDepartureDate (iOptimisationNotification._preferredDepartureDate)
      ,
00025       _notificationDateTime (iOptimisationNotification._notificationDateTime),
00026       _preferredCabin (iOptimisationNotification._preferredCabin),
00027       _partySize (iOptimisationNotification._partySize),
00028       _channel (iOptimisationNotification._channel),
00029       _tripType (iOptimisationNotification._tripType),
00030       _stayDuration (iOptimisationNotification._stayDuration),
00031       _frequentFlyerType (iOptimisationNotification._frequentFlyerType),
00032       _preferredDepartureTime (iOptimisationNotification._preferredDepartureTime)
      ,
00033       _wtp (iOptimisationNotification._wtp),
00034       _valueOfTime (iOptimisationNotification._valueOfTime) {
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   OptimisationNotificationStruct::
00039   OptimisationNotificationStruct (const AirportCode_T& iOrigin,
00040                         const AirportCode_T& iDestination,
00041                         const CityCode_T& iPOS,
00042                         const Date_T& iDepartureDate,
00043                         const DateTime_T& iNotificationDateTime,
00044                         const CabinCode_T& iPreferredCabin,
00045                         const NbOfSeats_T& iPartySize,
00046                         const ChannelLabel_T& iChannel,
00047                         const TripType_T& iTripType,
00048                         const DayDuration_T& iStayDuration,
00049                         const FrequentFlyer_T& iFrequentFlyerType,
00050                         const Duration_T& iPreferredDepartureTime,
00051                         const WTP_T& iWTP,
00052                         const PriceValue_T& iValueOfTime)
00053     : _origin (iOrigin), _destination (iDestination),
00054       _pos (iPOS), _preferredDepartureDate (iDepartureDate),
00055       _notificationDateTime (iNotificationDateTime),
00056       _preferredCabin (iPreferredCabin), _partySize (iPartySize),
00057       _channel (iChannel), _tripType (iTripType),
00058       _stayDuration (iStayDuration), _frequentFlyerType (iFrequentFlyerType),
00059       _preferredDepartureTime (iPreferredDepartureTime), _wtp (iWTP),
00060       _valueOfTime (iValueOfTime) {
00061   }
00062
```

```
// ////////////////////////////////////////////////////////////////////
OptimisationNotificationStruct::~OptimisationNotificationStruct() {
}

// ////////////////////////////////////////////////////////////////////
void OptimisationNotificationStruct::toStream (std::ostream& ioOut) const {
  ioOut << describe();
}

// ////////////////////////////////////////////////////////////////////
void OptimisationNotificationStruct::fromStream (std::istream& ioIn) {
}

// ////////////////////////////////////////////////////////////////////
const std::string OptimisationNotificationStruct::describe() const {
  std::ostringstream oStr;
  oStr << "At " << _notificationDateTime
       << ", for (" << _pos << ") " << _origin << "-" << _destination
       << " " << _preferredDepartureDate << " " << _preferredCabin
       << " " << _partySize << " " << _channel << " " << _tripType
       << " " << _stayDuration << " " << _frequentFlyerType
       << " " << _preferredDepartureTime << " " << _wtp
       << " " << _valueOfTime;
  return oStr.str();
}

}
```

33.401 stdair/bom/OptimisationNotificationStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>
```

Classes

- struct stdair::OptimisationNotificationStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.402 stdair/bom/OptimisationNotificationStruct.hpp

```
#ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
#define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/OptimisationNotificationTypes.hpp>

namespace stdair {

  struct OptimisationNotificationStruct : public StructAbstract {
  public:
    // /////////////// Getters /////////////////
    const AirportCode_T& getOrigin() const {
      return _origin;
    }

    const AirportCode_T& getDestination() const {
      return _destination;
    }

    const CityCode_T& getPOS() const {
      return _pos;
    }

    const Date_T& getPreferedDepartureDate() const {
      return _preferredDepartureDate;
    }

    const DateTime_T& getNotificationDateTime() const {
      return _notificationDateTime;
    }

    const CabinCode_T& getPreferredCabin() const {
      return _preferredCabin;
    }

    const NbOfSeats_T& getPartySize() const {
      return _partySize;
    }

    const ChannelLabel_T& getOptimisationChannel() const {
      return _channel;
    }

    const TripType_T& getTripType() const {
      return _tripType;
    }

    const DayDuration_T& getStayDuration() const {
      return _stayDuration;
    }

    const FrequentFlyer_T& getFrequentFlyerType() const {
      return _frequentFlyerType;
    }

    const Duration_T& getPreferredDepartureTime() const {
```

```
00079       return _preferredDepartureTime;
00080     }
00081
00083     const WTP_T& getWTP() const {
00084       return _wtp;
00085     }
00086
00088     const PriceValue_T& getValueOfTime () const {
00089       return _valueOfTime;
00090     }
00091
00092     // /////////// Display support method /////////////
00095     void toStream (std::ostream& ioOut) const;
00096
00099     void fromStream (std::istream& ioIn);
00100
00102     const std::string describe() const;
00103
00104
00105     // /////////////// Constructors and Destructors /////////////////
00106   public:
00108     OptimisationNotificationStruct (const AirportCode_T& iOrigin,
00109                                     const AirportCode_T& iDestination,
00110                                     const CityCode_T& iPOS,
00111                                     const Date_T& iDepartureDate,
00112                                     const DateTime_T& iNotificationDateTime,
00113                                     const CabinCode_T& iPreferredCabin,
00114                                     const NbOfSeats_T& iPartySize,
00115                                     const ChannelLabel_T& iChannel,
00116                                     const TripType_T& iTripType,
00117                                     const DayDuration_T& iStayDuration,
00118                                     const FrequentFlyer_T& iFrequentFlyerType,
00119                                     const Duration_T& iPreferredDepartureTime,
00120                                     const WTP_T& iWTP,
00121                                     const PriceValue_T& iValueOfTime);
00122
00124     OptimisationNotificationStruct (const OptimisationNotificationStruct&);
00125
00126   private:
00129     OptimisationNotificationStruct ();
00130
00131   public:
00133     ~OptimisationNotificationStruct();
00134
00135
00136   private:
00137     // /////////////// Attributes /////////////////
00139     const AirportCode_T _origin;
00140
00142     const AirportCode_T _destination;
00143
00145     const CityCode_T _pos;
00146
00148     const Date_T _preferredDepartureDate;
00149
00151     const DateTime_T _notificationDateTime;
00152
00154     const CabinCode_T _preferredCabin;
00155
00157     const NbOfSeats_T _partySize;
00158
00160     const ChannelLabel_T _channel;
00161
00164     const TripType_T _tripType;
00165
00167     const DayDuration_T _stayDuration;
00168
```

```
    const FrequentFlyer_T _frequentFlyerType;

    const Duration_T _preferredDepartureTime;

    const WTP_T _wtp;

    const PriceValue_T _valueOfTime;
  };

}
#endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONSTRUCT_HPP
```

33.403 stdair/bom/OptimisationNotificationTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< OptimisationNotificationStruct > stdair::OptimisationNotificationPtr_T

33.404 stdair/bom/OptimisationNotificationTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP
#define __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// Boost
#include <boost/shared_ptr.hpp>

namespace stdair {

  // Forward declarations
  struct OptimisationNotificationStruct;

  // ///////////// Type definitions //////////////
  typedef boost::
  shared_ptr<OptimisationNotificationStruct> OptimisationNotificationPtr_T;

}
#endif // __STDAIR_BOM_OPTIMISATIONNOTIFICATIONTYPES_HPP

```

33.405 stdair/bom/ParsedKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/tokenizer.hpp>
#include <boost/lexical_cast.hpp>
#include <boost/date_time/gregorian/parsers.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/InventoryKey.hpp>
#include <stdair/bom/FlightDateKey.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/LegDateKey.hpp>
#include <stdair/bom/ParsedKey.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- const boost::char_separator< char > stdair::TokeniserDashSeparator ("-")
- const boost::char_separator< char > stdair::TokeniserTimeSeparator (":")

33.406 stdair/bom/ParsedKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost
00008 #include <boost/tokenizer.hpp>
00009 #include <boost/lexical_cast.hpp>
00010 #include <boost/date_time/gregorian/parsers.hpp>
00011 // StdAir
00012 #include <stdair/stdair_exceptions.hpp>
00013 #include <stdair/basic/BasConst_Inventory.hpp>
00014 #include <stdair/basic/BasConst_BomDisplay.hpp>
00015 #include <stdair/bom/InventoryKey.hpp>
00016 #include <stdair/bom/FlightDateKey.hpp>
00017 #include <stdair/bom/SegmentDateKey.hpp>
00018 #include <stdair/bom/LegDateKey.hpp>
00019 #include <stdair/bom/ParsedKey.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024   // ////////////// Tokenising support ///////////////
00028   typedef boost::tokenizer<boost::char_separator<char> > Tokeniser_T;
00029
00033   const boost::char_separator<char> TokeniserDashSeparator ("-");
00034
00038   const boost::char_separator<char> TokeniserTimeSeparator (":");
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   ParsedKey::ParsedKey() : _fullKey (""), _airlineCode (""), _flightNumber (""),
00042                            _departureDate (""), _boardingPoint (""),
00043                            _offPoint (""), _boardingTime (="") {
00044   }
00045
00046   // ////////////////////////////////////////////////////////////////////
00047   ParsedKey::~ParsedKey() {
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   InventoryKey ParsedKey::getInventoryKey() const {
00052     if (_airlineCode.size() < 2 || _airlineCode.size() > 3) {
00053       STDAIR_LOG_ERROR ("No airline code can be found in '" << _fullKey << "'");
00054       STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00055       throw KeyNotFoundException ("No airline code can be found in '"
00056                                   + _fullKey + "'");
00057     }
00058     return _airlineCode;
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   FlightDateKey ParsedKey::getFlightDateKey() const {
00063     // Check whether the departure date has been parsed correctly.
00064     Tokeniser_T lDateTokens (_departureDate, TokeniserDashSeparator);
00065
00066     if (lDateTokens.begin() == lDateTokens.end()) {
00067       STDAIR_LOG_ERROR ("No date can be found in '" << _fullKey << "'");
00068       STDAIR_LOG_DEBUG ("Parsed key: " << toString());
00069       throw KeyNotFoundException ("No date can be found in '" + _fullKey + "'");
00070     }
00071
00072     const FlightNumber_T lFlightNumber =
00073       boost::lexical_cast<FlightNumber_T> (_flightNumber);
00074
```

```
    const Date_T lDepartureDate =
      boost::gregorian::from_simple_string (_departureDate);

    const FlightDateKey oFlightDateKey (lFlightNumber, lDepartureDate);

    return oFlightDateKey;
  }

  // ////////////////////////////////////////////////////////////////////
  LegDateKey ParsedKey::getLegKey() const {
    if (_boardingPoint.size() != 3) {
      STDAIR_LOG_ERROR ("No airport code can be found in '" << _fullKey << "'");
      STDAIR_LOG_DEBUG ("Parsed key: " << toString());
      throw KeyNotFoundException ("No airport code can be found in '"
                                  + _fullKey + "'");
    }

    const LegDateKey oLegDateKey (_boardingPoint);

    return oLegDateKey;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDateKey ParsedKey::getSegmentKey() const {
    if (_boardingPoint.size() != 3 || _offPoint.size() != 3) {
      STDAIR_LOG_ERROR ("No airport code can be found in '" << _fullKey << "'");
      STDAIR_LOG_DEBUG ("Parsed key: " << toString());
      throw KeyNotFoundException ("No airport code can be found in '"
                                  + _fullKey + "'");
    }

    const SegmentDateKey oSegmentDateKey (_boardingPoint, _offPoint);

    return oSegmentDateKey;
  }

  // ////////////////////////////////////////////////////////////////////
  const Duration_T ParsedKey::getBoardingTime() const {
    // Check whether the boarding time has been parsed correctly.
    Tokeniser_T lTimeTokens (_boardingTime, TokeniserTimeSeparator);

    if (lTimeTokens.begin() == lTimeTokens.end()) {
      STDAIR_LOG_ERROR ("No boarding time can be found in '" << _fullKey << "'");

      STDAIR_LOG_DEBUG ("Parsed key: " << toString());
      throw KeyNotFoundException ("No boarding time can be found in '"
                                  + _fullKey + "'");
    }

    const Duration_T oBoardingTime (boost::posix_time::
                                    duration_from_string (_boardingTime));

    return oBoardingTime;
  }

  // ////////////////////////////////////////////////////////////////////
  void ParsedKey::toStream (std::ostream& ioOut) const {
    ioOut << "ParsedKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void ParsedKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string ParsedKey::toString() const {
    std::ostringstream oStr;
```

```

    oStr << _airlineCode
         << DEFAULT_KEY_FLD_DELIMITER << " "
         << _flightNumber
         << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
         << _departureDate
         << DEFAULT_KEY_FLD_DELIMITER << " "
         << _boardingPoint
         << DEFAULT_KEY_SUB_FLD_DELIMITER << " "
         << _offPoint
         << DEFAULT_KEY_FLD_DELIMITER << " "
         << _boardingTime;

    return oStr.str();
  }

}
```

33.407 stdair/bom/ParsedKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::ParsedKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.408 stdair/bom/ParsedKey.hpp

```
#ifndef __STDAIR_BOM_PARSEDKEY_HPP
#define __STDAIR_BOM_PARSEDKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace stdair {
  struct InventoryKey;
  struct FlightDateKey;
  struct SegmentDateKey;
  struct LegDateKey;

  struct ParsedKey : public KeyAbstract{

    // ////////////// Getter ///////////////
    InventoryKey getInventoryKey () const;

    FlightDateKey getFlightDateKey () const;

    SegmentDateKey getSegmentKey () const;

    LegDateKey getLegKey () const;

    const Duration_T getBoardingTime () const;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

  public:
    // /////////////// Constructor and destructor. ///////////////
    // Default constructor
    ParsedKey ();
    // Defaut destructor
    ~ParsedKey ();

  public:
    // ////////////// Attributes ///////////////
    std::string _fullKey;
    std::string _airlineCode;
    std::string _flightNumber;
    std::string _departureDate;
    std::string _boardingPoint;
    std::string _offPoint;
    std::string _boardingTime;
  };

}
#endif // __STDAIR_BOM_PARSEDKEY_HPP
```

33.409 stdair/bom/PeriodStruct.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <stdair/basic/BasConst_Period_BOM.hpp>
#include <stdair/bom/PeriodStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.410 stdair/bom/PeriodStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Period_BOM.hpp>
00009 #include <stdair/bom/PeriodStruct.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   PeriodStruct::PeriodStruct ()
00015     : _dateRange (BOOST_DEFAULT_DATE_PERIOD), _dow () {
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   PeriodStruct::PeriodStruct (const DatePeriod_T& iDateRange,
00020                               const DoWStruct& iDoW)
00021     : _dateRange (iDateRange), _dow (iDoW) {
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   PeriodStruct::PeriodStruct (const PeriodStruct& iPeriodStruct)
00026     : _dateRange (iPeriodStruct._dateRange), _dow (iPeriodStruct._dow) {
00027
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   const std::string PeriodStruct::describeShort() const {
00032     std::ostringstream ostr;
00033     ostr << _dateRange << ", " << _dow.describeShort ();
00034     return ostr.str();
00035   }
00036
00037   // ////////////////////////////////////////////////////////////////////
00038   const std::string PeriodStruct::describe() const {
00039     std::ostringstream ostr;
00040     ostr << _dateRange << ", " << _dow.describe ();
00041     return ostr.str();
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   PeriodStruct PeriodStruct::
00046   addDateOffset (const DateOffset_T& iDateOffset) const {
00047     // Create a new date range by shifting the date range of this object with
00048     // iDateOffset.
00049     DatePeriod_T lNewDateRange = getDateRange();
00050     lNewDateRange.shift (iDateOffset);
00051
00052     // Create a new DoWStruct by shifting the DoWStruct of this object with
00053     // iDateOffset.
00054     const long lNbOfDaysOffset = iDateOffset.days();
00055     const DoWStruct& lDoW = getDoW();
00056     const DoWStruct lNewDoW = lDoW.shift (lNbOfDaysOffset);
00057
00058     return PeriodStruct (lNewDateRange, lNewDoW);
00059   }
00060
00061   // ////////////////////////////////////////////////////////////////////
00062   PeriodStruct PeriodStruct::
00063   intersection (const PeriodStruct& iPeriodStruct) const {
00064     const DatePeriod_T lNewDateRange =
00065       _dateRange.intersection (iPeriodStruct._dateRange);
```

```
    const DoWStruct lNewDoW = _dow.intersection (iPeriodStruct._dow);

    return PeriodStruct (lNewDateRange, lNewDoW);
  }

  // ////////////////////////////////////////////////////////////////////
  const bool PeriodStruct::isValid () const {
    if (_dateRange.is_null() == false && _dow.isValid()) {
      return true;
    }
    return false;
  }

}
```

33.411 stdair/bom/PeriodStruct.hpp File Reference

```
#include <string>
#include <vector>
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>
```

Classes

- struct stdair::PeriodStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.412 stdair/bom/PeriodStruct.hpp

```
#ifndef __STDAIR_BOM_PERIODSTRUCT_HPP
#define __STDAIR_BOM_PERIODSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/DoWStruct.hpp>

namespace stdair {

  struct PeriodStruct : public StructAbstract {
  public:
    // ////////// Getters /////////////
    const DatePeriod_T& getDateRange () const {
      return _dateRange;
    }
    const DoWStruct& getDoW () const {
      return _dow;
    }

  public:
    // /////////// Setters ////////////
    void setDateRange (const DatePeriod_T& iDateRange) {
      _dateRange = iDateRange;
    }
    void setDoW (const DoWStruct& iDoW) { _dow = iDoW; }

  public:
    const std::string describe() const;

    const std::string describeShort() const;

  public:
    // /////////// Business Methods /////////////
    PeriodStruct addDateOffset (const DateOffset_T&) const;

    PeriodStruct intersection (const PeriodStruct&) const;

    const bool isValid () const;

  public:
    PeriodStruct (const DatePeriod_T&, const DoWStruct&);
    PeriodStruct ();
    PeriodStruct (const PeriodStruct&);
    ~PeriodStruct () { }

  private:
    // Attributes
    DatePeriod_T _dateRange;
    DoWStruct _dow;
  };

}
#endif // __STDAIR_BOM_PERIODSTRUCT_HPP
```

33.413 stdair/bom/Policy.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/Policy.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.414 stdair/bom/Policy.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <sstream>
00006 #include <cassert>
00007 #include <iomanip>
00008 #include <iostream>
00009 //STDAIR
00010 #include <stdair/basic/BasConst_Inventory.hpp>
00011 #include <stdair/bom/BomManager.hpp>
00012 #include <stdair/bom/BookingClass.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/Policy.hpp>
00015
00016 namespace stdair {
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   Policy::Policy () :
00020     _key (DEFAULT_POLICY_CODE), _parent (NULL) {
00021     assert (false);
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   Policy::Policy (const Policy& iPolicy)
00026   : _key (DEFAULT_POLICY_CODE), _parent (NULL) {
00027     assert (false);
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   Policy::Policy (const Key_T& iKey) : _key (iKey), _parent (NULL) {
00032   }
00033
00034   // ////////////////////////////////////////////////////////////////////
00035   Policy::~Policy() {
00036   }
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   std::string Policy::toString () const {
00040     std::ostringstream oStr;
00041     oStr << describeKey();
00042
00043     oStr << std::fixed << std::setprecision (2)
00044          << "; " << _demand
00045          << "; " << _stdDev
00046          << "; " << _yield << std::endl;
00047
00048     return oStr.str();
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   const BookingClassList_T& Policy::getBookingClassList() const {
00053     return BomManager::getList<BookingClass> (*this);
00054   }
00055
00056   // ////////////////////////////////////////////////////////////////////
00057   const Revenue_T Policy::getTotalRevenue () const {
00058     Revenue_T oTotalRevenue = 0.0;
00059     for (YieldDemandMap_T::const_iterator itYD = _yieldDemandMap.begin();
00060          itYD != _yieldDemandMap.end(); ++itYD) {
00061       const Yield_T& lYield = itYD->first;
00062       const double& lDemand = itYD->second;
00063       oTotalRevenue += lYield*lDemand;
00064     }
00065
```

```
    return oTotalRevenue;
  }

  // ////////////////////////////////////////////////////////////////////
  void Policy::addYieldDemand (const Yield_T& iYield,
                               const NbOfBookings_T& iDemand) {
    YieldDemandMap_T::iterator itYD = _yieldDemandMap.find (iYield);
    if (itYD == _yieldDemandMap.end()) {
        bool insert = _yieldDemandMap.insert (YieldDemandMap_T::value_type
                                              (iYield, iDemand)).second;
        assert (insert == true);
    } else {
      NbOfBookings_T& lDemand = itYD->second;
      lDemand += iDemand;
    }
  }

}
```

33.415 stdair/bom/Policy.hpp File Reference

```
#include <cmath>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/PolicyKey.hpp>
```

Classes

- class stdair::Policy

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.416 stdair/bom/Policy.hpp

```
00001 #ifndef __STDAIR_BOM_POLICY_HPP
00002 #define __STDAIR_BOM_POLICY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <cmath>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/stdair_rm_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/BookingClassTypes.hpp>
00014 #include <stdair/bom/PolicyKey.hpp>
00015
00017 namespace boost {
00018   namespace serialization {
00019     class access;
00020   }
00021 }
00022
00023 namespace stdair {
00024
00030   class Policy : public BomAbstract {
00031     template <typename BOM> friend class FacBom;
00032     friend class FacBomManager;
00033     friend class boost::serialization::access;
00034
00035   public:
00036     // ////////// Type definitions ////////////
00040     typedef PolicyKey Key_T;
00041
00042   public:
00043     // ////////////////// Getters ////////////////////////
00045     const Key_T& getKey() const {
00046       return _key;
00047     }
00048
00050     BomAbstract* const getParent() const {
00051       return _parent;
00052     }
00053
00057     const HolderMap_T& getHolderMap() const {
00058       return _holderMap;
00059     }
00060
00062     const BookingClassList_T& getBookingClassList() const;
00063
00065     const NbOfBookings_T& getDemand() const {
00066       return _demand;
00067     }
00068
00070     const StdDevValue_T& getStdDev() const {
00071       return _stdDev;
00072     }
00073
00075     const Yield_T& getYield() const {
00076       return _yield;
00077     }
00078
00080     const Revenue_T getTotalRevenue () const;
00081
00082   public:
00083     // ////////////////////// Setters ////////////////////
00085     void setDemand (const NbOfBookings_T& iDemand) {
```

```
00086       _demand = iDemand;
00087     }
00088
00090     void setStdDev (const StdDevValue_T& iStdDev) {
00091       _stdDev = iStdDev;
00092     }
00093
00095     void setYield (const Yield_T& iYield) {
00096       _yield = iYield;
00097     }
00098
00100     void resetDemandForecast () {
00101       _demand = 0.0;
00102       _stdDev = 0.0;
00103       _yieldDemandMap.clear();
00104     }
00105
00107     void addYieldDemand (const Yield_T&, const NbOfBookings_T&);
00108
00109   public:
00110     // /////////// Display support methods /////////
00116     void toStream (std::ostream& ioOut) const {
00117       ioOut << toString();
00118     }
00119
00125     void fromStream (std::istream& ioIn) {
00126     }
00127
00131     std::string toString() const;
00132
00136     const std::string describeKey() const {
00137       return _key.toString();
00138     }
00139
00140
00141   public:
00142     // /////////// (Boost) Serialisation support methods /////////
00146     template<class Archive>
00147     void serialize (Archive& ar, const unsigned int iFileVersion);
00148
00149   private:
00157     void serialisationImplementationExport() const;
00158     void serialisationImplementationImport();
00159
00160
00161   protected:
00162     // /////////// Constructors and destructor. ////////////
00166     Policy (const Key_T&);
00167
00171     virtual ~Policy();
00172
00173   private:
00177     Policy();
00178
00182     Policy (const Policy&);
00183
00184
00185   private:
00186     // ///////////// Attributes ////////////
00190     Key_T _key;
00191
00195     BomAbstract* _parent;
00196
00200     HolderMap_T _holderMap;
00201
00205     NbOfBookings_T _demand;
00206
```

```
    StdDevValue_T _stdDev;

    Yield_T _yield;

    YieldDemandMap_T _yieldDemandMap;

  };
}
#endif // __STDAIR_BOM_POLICY_HPP
```

33.417 stdair/bom/PolicyKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/PolicyKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::PolicyKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::PolicyKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.418 stdair/bom/PolicyKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/PolicyKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   PolicyKey::PolicyKey() : _policyCode (DEFAULT_POLICY_CODE) {
00019     assert (false);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   PolicyKey::PolicyKey (const PolicyKey& iPolicyKey)
00024     : _policyCode (iPolicyKey._policyCode) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   PolicyKey::PolicyKey (const PolicyCode_T& iPolicyCode)
00029     : _policyCode (iPolicyCode) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   PolicyKey::~PolicyKey() {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void PolicyKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "PolicyKey: " << toString();
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void PolicyKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string PolicyKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _policyCode;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void PolicyKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void PolicyKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void PolicyKey::serialize (Archive& ioArchive,
00069                              const unsigned int iFileVersion) {
00074     ioArchive & _policyCode;
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   // Explicit template instantiation
00079   namespace ba = boost::archive;
00080   template void PolicyKey::serialize<ba::text_oarchive> (ba::text_oarchive&,
00081                                                          unsigned int);
00082   template void PolicyKey::serialize<ba::text_iarchive> (ba::text_iarchive&,
00083                                                          unsigned int);
00084   // ////////////////////////////////////////////////////////////////////
00085
00086 }
```

33.419 stdair/bom/PolicyKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::PolicyKey

  Key of a given policy, made of a policy code.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.420 stdair/bom/PolicyKey.hpp

```
00001 #ifndef __STDAIR_BOM_POLICYKEY_HPP
00002 #define __STDAIR_BOM_POLICYKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct PolicyKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     PolicyKey();
00035
00036   public:
00040     PolicyKey (const PolicyCode_T& iPolicyCode);
00041
00045     PolicyKey (const PolicyKey&);
00046
00050     ~PolicyKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const PolicyCode_T& getPolicyCode () const {
00057       return _policyCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     PolicyCode_T _policyCode;
```

```
00112   };
00113
00114 }
00115 #endif // __STDAIR_BOM_POLICYKEY_HPP
```

33.421 stdair/bom/PolicyTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< Policy * > stdair::PolicyList_T
- typedef std::map< const MapKey_T, Policy * > stdair::PolicyMap_T

33.422 stdair/bom/PolicyTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_POLICYTYPES_HPP
#define __STDAIR_BOM_POLICYTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class Policy;

  typedef std::list<Policy*> PolicyList_T;

  typedef std::map<const MapKey_T, Policy*> PolicyMap_T;

}
#endif // __STDAIR_BOM_POLICYTYPES_HPP
```

33.423 stdair/bom/PosChannel.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/PosChannel.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.424 stdair/bom/PosChannel.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/PosChannel.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  PosChannel::PosChannel()
    : _key (DEFAULT_POS,
            DEFAULT_CHANNEL),
      _parent (NULL)  {
    // That constructor is used by the serialisation process
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannel::PosChannel (const PosChannel& iPosChannel)
    : _key (iPosChannel.getKey()), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannel::PosChannel (const Key_T& iKey)
    : _key (iKey), _parent (NULL)  {
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannel::~PosChannel () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string PosChannel::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }
}
```

33.425 stdair/bom/PosChannel.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/PosChannelKey.hpp>
#include <stdair/bom/PosChannelTypes.hpp>
```

Classes

- class stdair::PosChannel

  Class representing the actual attributes for a fare point of sale.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.426 stdair/bom/PosChannel.hpp

```
#ifndef __STDAIR_BOM_POSCHANNEL_HPP
#define __STDAIR_BOM_POSCHANNEL_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/PosChannelKey.hpp>
#include <stdair/bom/PosChannelTypes.hpp>

// Forward declaration
namespace stdair {

  class PosChannel : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;

  public:
    // /////// Type definitions
    typedef PosChannelKey Key_T;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }

  public:
    // ////////// Getters ////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const  stdair::HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const CityCode_T& getPos() const {
      return _key.getPos();
    }

    const ChannelLabel_T& getChannel() const {
      return _key.getChannel();
    }

  protected:
    // ////////// Constructors and destructors /////////
    PosChannel (const Key_T&);

    virtual ~PosChannel();
```

```
00110
00111   private:
00115     PosChannel ();
00116
00120     PosChannel (const PosChannel&);
00121
00122   protected:
00123     // ////////// Attributes ////////
00127     Key_T _key;
00128
00132     BomAbstract* _parent;
00133
00137     HolderMap_T _holderMap;
00138
00139   };
00140
00141 }
00142 #endif // __STDAIR_BOM_POSCHANNEL_HPP
00143
```

33.427 stdair/bom/PosChannelKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/PosChannelKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.428 stdair/bom/PosChannelKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// STDAIR
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/PosChannelKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  PosChannelKey::PosChannelKey()
    : _pos (DEFAULT_POS),
      _channel (DEFAULT_CHANNEL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannelKey::PosChannelKey (const CityCode_T& iPos,
                                const ChannelLabel_T& iChannel)
    : _pos (iPos), _channel(iChannel) {
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannelKey::PosChannelKey (const PosChannelKey& iKey)
    : _pos (iKey._pos), _channel (iKey._channel) {
  }

  // ////////////////////////////////////////////////////////////////////
  PosChannelKey::~PosChannelKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void PosChannelKey::toStream (std::ostream& ioOut) const {
    ioOut << "PosChannelKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void PosChannelKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string PosChannelKey::toString() const {
    std::ostringstream oStr;
    oStr << _pos << DEFAULT_KEY_SUB_FLD_DELIMITER
         << " " << _channel;
    return oStr.str();
  }

}
```

33.429 stdair/bom/PosChannelKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_types.hpp>
```

Classes

- struct stdair::PosChannelKey

  Key of point of sale and channel.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.430 stdair/bom/PosChannelKey.hpp

```
00001 #ifndef __STDAIR_BOM_POSCHANNELKEY_HPP
00002 #define __STDAIR_BOM_POSCHANNELKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // SIMFQT
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_types.hpp>
00010
00011 namespace stdair  {
00015   struct PosChannelKey : public KeyAbstract {
00016
00017   public:
00018     // /////////// Construction ///////////
00022     PosChannelKey (const stdair::CityCode_T&, const stdair::ChannelLabel_T&);
00026     PosChannelKey (const PosChannelKey&);
00030     ~PosChannelKey ();
00031   private:
00035     PosChannelKey ();
00036
00037   public:
00038     // /////////// Getters //////////
00039
00043     const stdair::CityCode_T& getPos() const {
00044       return _pos;
00045     }
00046
00050     const stdair::ChannelLabel_T& getChannel() const {
00051       return _channel;
00052     }
00053
00054   public:
00055     // /////////// Display support methods /////////
00060     void toStream (std::ostream& ioOut) const;
00061
00066     void fromStream (std::istream& ioIn);
00067
00072     const std::string toString() const;
00073
00074   private:
00075     // /////////////// Attributes /////////////////
00079     CityCode_T _pos;
00080
00085     ChannelLabel_T _channel;
00086
00087   };
00088
00089 }
00090 #endif // __STDAIR_BOM_POSCHANNELKEY_HPP
```

33.431 stdair/bom/PosChannelTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< PosChannel * > stdair::PosChannelList_T
- typedef std::map< const MapKey_T, PosChannel * > stdair::PosChannelMap_T
- typedef std::pair< MapKey_T, PosChannel * > stdair::PosChannelWithKey_T
- typedef std::list< PosChannelWithKey_T > stdair::PosChannelDetailedList_T

33.432 stdair/bom/PosChannelTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_POSCHANNELTYPES_HPP
#define __STDAIR_BOM_POSCHANNELTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class PosChannel;

  typedef std::list<PosChannel*> PosChannelList_T;

  typedef std::map<const MapKey_T, PosChannel*> PosChannelMap_T;

  typedef std::pair<MapKey_T, PosChannel*> PosChannelWithKey_T;
  typedef std::list<PosChannelWithKey_T> PosChannelDetailedList_T;
}
#endif // __STDAIR_BOM_POSCHANNELTYPES_HPP
```

33.433 stdair/bom/RMEventStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/RMEventStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.434 stdair/bom/RMEventStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/RMEventStruct.hpp>
00009
00010 namespace stdair {
00011
00012   // ////////////////////////////////////////////////////////////////////
00013   RMEventStruct::RMEventStruct() {
00014     assert (false);
00015   }
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   RMEventStruct::
00019   RMEventStruct (const RMEventStruct& iRMEvent)
00020     : _airlineCode (iRMEvent._airlineCode),
00021       _flightDateDescription (iRMEvent._flightDateDescription),
00022       _RMEventTime (iRMEvent._RMEventTime) {
00023   }
00024
00025   // ////////////////////////////////////////////////////////////////////
00026   RMEventStruct::
00027   RMEventStruct (const AirlineCode_T& iAirlineCode,
00028                  const KeyDescription_T& iFlightDateDescription,
00029                  const DateTime_T& iRMEventTime)
00030     : _airlineCode (iAirlineCode),
00031       _flightDateDescription (iFlightDateDescription),
00032       _RMEventTime (iRMEventTime) {
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   RMEventStruct::~RMEventStruct() {
00037   }
00038
00039   // ////////////////////////////////////////////////////////////////////
00040   void RMEventStruct::toStream (std::ostream& ioOut) const {
00041     ioOut << describe();
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   void RMEventStruct::fromStream (std::istream& ioIn) {
00046   }
00047
00048   // ////////////////////////////////////////////////////////////////////
00049   const std::string RMEventStruct::describe() const {
00050     std::ostringstream oStr;
00051     oStr << _airlineCode << ", " << _flightDateDescription << ", "
00052          << _RMEventTime;
00053     return oStr.str();
00054   }
00055
00056 }
```

33.435 stdair/bom/RMEventStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/RMEventTypes.hpp>
```

Classes

- struct stdair::RMEventStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.436 stdair/bom/RMEventStruct.hpp

```
00001 #ifndef __STDAIR_BOM_RMEVENTSTRUCT_HPP
00002 #define __STDAIR_BOM_RMEVENTSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/RMEventTypes.hpp>
00015
00016 namespace stdair {
00017
00019   struct RMEventStruct : public StructAbstract {
00020   public:
00021     // /////////////// Getters /////////////////
00023     const AirlineCode_T& getAirlineCode() const {
00024       return _airlineCode;
00025     }
00026
00028     const KeyDescription_T& getFlightDateDescription() const {
00029       return _flightDateDescription;
00030     }
00031
00033     const DateTime_T& getRMEventTime() const {
00034       return _RMEventTime;
00035     }
00036
00037     // /////////// Display support method /////////////
00040     void toStream (std::ostream& ioOut) const;
00041
00044     void fromStream (std::istream& ioIn);
00045
00047     const std::string describe() const;
00048
00049
00050     // /////////////// Constructors and Destructors /////////////////
00051   public:
00053     RMEventStruct (const AirlineCode_T&, const KeyDescription_T&,
00054                    const DateTime_T&);
00055
00057     RMEventStruct (const RMEventStruct&);
00058
00061     RMEventStruct ();
00062
00063   public:
00065     ~RMEventStruct();
00066
00067
00068   private:
00069     // /////////////// Attributes /////////////////
00071     const AirlineCode_T _airlineCode;
00072
00074     const KeyDescription_T _flightDateDescription;
00075
00077     const DateTime_T _RMEventTime;
00078   };
00079
00080 }
00081 #endif // __STDAIR_BOM_RMEVENTSTRUCT_HPP
```

33.437 stdair/bom/RMEventTypes.hpp File Reference

```
#include <list>
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< RMEventStruct > stdair::RMEventPtr_T
- typedef std::list< RMEventStruct > stdair::RMEventList_T

33.438 stdair/bom/RMEventTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_RMEVENTTYPES_HPP
00003 #define __STDAIR_BOM_RMEVENTTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <list>
00010 // Boost
00011 #include <boost/shared_ptr.hpp>
00012
00013 namespace stdair {
00014
00015   // Forward declarations
00016   struct RMEventStruct;
00017
00018   // ///////////// Type definitions //////////////
00020   typedef boost::shared_ptr<RMEventStruct> RMEventPtr_T;
00021
00023   typedef std::list<RMEventStruct> RMEventList_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_RMEVENTTYPES_HPP
00027
```

33.439 stdair/bom/SegmentCabin.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_Yield.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/Policy.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.440 stdair/bom/SegmentCabin.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/basic/BasConst_Inventory.hpp>
00010 #include <stdair/basic/BasConst_Yield.hpp>
00011 #include <stdair/basic/BasConst_BomDisplay.hpp>
00012 #include <stdair/bom/BomManager.hpp>
00013 #include <stdair/bom/SegmentDate.hpp>
00014 #include <stdair/bom/SegmentCabin.hpp>
00015 #include <stdair/bom/BookingClass.hpp>
00016 #include <stdair/bom/BookingClassTypes.hpp>
00017 #include <stdair/bom/Policy.hpp>
00018
00019 namespace stdair {
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   SegmentCabin::SegmentCabin() : _key (DEFAULT_CABIN_CODE), _parent (NULL) {
00023     assert (false);
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   SegmentCabin::SegmentCabin (const SegmentCabin& iSegmentCabin)
00028     : _key (iSegmentCabin._key), _parent (NULL),
00029       _capacity (iSegmentCabin._capacity),
00030       _blockSpace (iSegmentCabin._blockSpace),
00031       _bookingCounter (iSegmentCabin._bookingCounter),
00032       _committedSpace (iSegmentCabin._committedSpace),
00033       _availabilityPool (iSegmentCabin._availabilityPool),
00034       _currentBidPrice (iSegmentCabin._currentBidPrice),
00035       _fareFamilyActivation (iSegmentCabin._fareFamilyActivation) {
00036   }
00037
00038   // ////////////////////////////////////////////////////////////////////
00039   SegmentCabin::SegmentCabin (const Key_T& iKey)
00040     : _key (iKey), _parent (NULL),
00041       _capacity (DEFAULT_CABIN_CAPACITY),
00042       _blockSpace (DEFAULT_BLOCK_SPACE),
00043       _bookingCounter (DEFAULT_CLASS_NB_OF_BOOKINGS),
00044       _committedSpace (DEFAULT_COMMITTED_SPACE),
00045       _availabilityPool (DEFAULT_AVAILABILITY),
00046       _bidPriceVector (DEFAULT_BID_PRICE_VECTOR),
00047       _currentBidPrice (DEFAULT_BID_PRICE),
00048       _fareFamilyActivation (false) {
00049   }
00050
00051   // ////////////////////////////////////////////////////////////////////
00052   SegmentCabin::~SegmentCabin() {
00053   }
00054
00055   // ////////////////////////////////////////////////////////////////////
00056   const MapKey_T SegmentCabin::getFullerKey() const {
00057     const SegmentDate& lSegmentDate = BomManager::getParent<SegmentDate>(*this);
00058
00059     const MapKey_T oFullKey =
00060       lSegmentDate.describeKey() + DEFAULT_KEY_FLD_DELIMITER + getCabinCode();
00061     return oFullKey;
00062   }
00063
00064   // ////////////////////////////////////////////////////////////////////
00065   std::string SegmentCabin::toString() const {
```

```
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SegmentCabin::describeConvexHull() const{
    std::ostringstream oStr;
    for (PolicyList_T::const_iterator itP = _convexHull.begin();
        itP != _convexHull.end(); ++itP) {
      const Policy* lPolicy = *itP;
      assert (lPolicy != NULL);
      oStr << lPolicy->toString();
    }
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentCabin::
  updateFromReservation (const NbOfBookings_T& iNbOfBookings) {
    _committedSpace += iNbOfBookings;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentCabin::addPolicy (Policy& ioPolicy) {
    _convexHull.push_back (&ioPolicy);
  }
}

```

33.441 stdair/bom/SegmentCabin.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
#include <stdair/bom/SegmentCabinTypes.hpp>
#include <stdair/bom/PolicyTypes.hpp>
```

Classes

- class stdair::SegmentCabin

  Class representing the actual attributes for an airline segment-cabin.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.442 stdair/bom/SegmentCabin.hpp

```
#ifndef __STDAIR_BOM_SEGMENTCABIN_HPP
#define __STDAIR_BOM_SEGMENTCABIN_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
#include <stdair/bom/SegmentCabinTypes.hpp>
#include <stdair/bom/PolicyTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {
  // Forward declarations
  class SegmentSnapshotTable;
  class Policy;

  class SegmentCabin : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef SegmentCabinKey Key_T;


  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const CabinCode_T& getCabinCode() const {
      return _key.getCabinCode();
    }

    const MapKey_T getFullerKey() const;

    const SegmentSnapshotTable& getSegmentSnapshotTable() const {
      assert (_segmentSnapshotTable != NULL);
      return *_segmentSnapshotTable;
    }

    const CabinCapacity_T& getCapacity() const {
      return _capacity;
```

```
    }

    const BlockSpace_T& getBlockSpace() const {
      return _blockSpace;
    }

    const BlockSpace_T& getMIN() const {
      return _min;
    }

    const UPR_T& getUPR() const {
      return _upr;
    }

    const NbOfBookings_T& getBookingCounter() const {
      return _bookingCounter;
    }

    const CommittedSpace_T& getCommittedSpace() const {
      return _committedSpace;
    }

    const Availability_T& getAvailabilityPool() const {
      return _availabilityPool;
    }

    const BidPrice_T& getCurrentBidPrice() const {
      return _currentBidPrice;
    }

    const BidPriceVector_T& getBidPriceVector() const {
      return _bidPriceVector;
    }

    const bool getFareFamilyStatus() const {
      return _fareFamilyActivation;
    }

    const PolicyList_T& getConvexHull() const {
      return _convexHull;
    }

  public:
    // ///////// Setters //////////
    void setSegmentSnapshotTable (SegmentSnapshotTable& ioTable) {
      _segmentSnapshotTable = &ioTable;
    }

    void setCapacity (const CabinCapacity_T& iCapacity) {
      _capacity = iCapacity;
    }

    void setBlockSpace (const BlockSpace_T& iBlockSpace) {
      _blockSpace = iBlockSpace;
    }

    void setMIN (const BlockSpace_T& iMIN) {
      _min = iMIN;
    }

    void setUPR (const UPR_T& iUPR) {
      _upr = iUPR;
    }

    void setBookingCounter (const NbOfBookings_T& iBookingCounter) {
      _bookingCounter = iBookingCounter;
    }
```

```
00179
00181     void setCommittedSpace (const CommittedSpace_T& iCommittedSpace) {
00182       _committedSpace = iCommittedSpace;
00183     }
00184
00186     void setAvailabilityPool (const Availability_T& iAvailabilityPool) {
00187       _availabilityPool = iAvailabilityPool;
00188     }
00189
00191     void setBidPriceVector (const BidPriceVector_T& iBPV) {
00192       _bidPriceVector = iBPV;
00193     }
00194
00196     void activateFareFamily () {
00197       _fareFamilyActivation = true;
00198     }
00199
00200   public:
00201     // /////////// Business methods //////////
00203     void updateFromReservation (const NbOfBookings_T&);
00204
00206     void resetConvexHull () { _convexHull.clear(); }
00207
00212     void addPolicy (Policy&);
00213
00214   public:
00215     // /////////// Display support methods /////////
00221     void toStream (std::ostream& ioOut) const {
00222       ioOut << toString();
00223     }
00224
00230     void fromStream (std::istream& ioIn) {
00231     }
00232
00236     std::string toString() const;
00237
00241     const std::string describeKey() const {
00242       return _key.toString();
00243     }
00244
00248     const std::string describeConvexHull() const;
00249
00250
00251   public:
00252     // /////////// (Boost) Serialisation support methods /////////
00256     template<class Archive>
00257     void serialize (Archive& ar, const unsigned int iFileVersion);
00258
00259   private:
00267     void serialisationImplementationExport() const;
00268     void serialisationImplementationImport();
00269
00270
00271   protected:
00272     // ////////// Constructors and destructors /////////
00276     SegmentCabin (const Key_T&);
00277
00281     virtual ~SegmentCabin();
00282
00283   private:
00287     SegmentCabin();
00288
00292     SegmentCabin (const SegmentCabin&);
00293
00294
00295   protected:
00296     // ////////// Attributes /////////
```

```
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;

    SegmentSnapshotTable* _segmentSnapshotTable;

    CabinCapacity_T _capacity;

    BlockSpace_T _blockSpace;

    BlockSpace_T _min;

    UPR_T _upr;

    NbOfBookings_T _bookingCounter;

    CommittedSpace_T _committedSpace;

    Availability_T _availabilityPool;

    BidPriceVector_T _bidPriceVector;

    BidPrice_T _currentBidPrice;

    bool _fareFamilyActivation;

    PolicyList_T _convexHull;

  };

}
#endif // __STDAIR_BOM_SEGMENTCABIN_HPP
```

33.443 stdair/bom/SegmentCabinKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/SegmentCabinKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::SegmentCabinKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentCabinKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.444 stdair/bom/SegmentCabinKey.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/access.hpp>
00011 // StdAir
00012 #include <stdair/basic/BasConst_Inventory.hpp>
00013 #include <stdair/bom/SegmentCabinKey.hpp>
00014
00015 namespace stdair {
00016
00017   // ////////////////////////////////////////////////////////////////////
00018   SegmentCabinKey::SegmentCabinKey() : _cabinCode (DEFAULT_CABIN_CODE) {
00019     assert (false);
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   SegmentCabinKey::SegmentCabinKey (const CabinCode_T& iCabinCode)
00024     : _cabinCode (iCabinCode) {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   SegmentCabinKey::SegmentCabinKey (const SegmentCabinKey& iKey)
00029     : _cabinCode (iKey._cabinCode) {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   SegmentCabinKey::~SegmentCabinKey () {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   void SegmentCabinKey::toStream (std::ostream& ioOut) const {
00038     ioOut << "SegmentCabinKey: " << toString();
00039   }
00040
00041   // ////////////////////////////////////////////////////////////////////
00042   void SegmentCabinKey::fromStream (std::istream& ioIn) {
00043   }
00044
00045   // ////////////////////////////////////////////////////////////////////
00046   const std::string SegmentCabinKey::toString() const {
00047     std::ostringstream oStr;
00048     oStr << _cabinCode;
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   void SegmentCabinKey::serialisationImplementationExport() const {
00054     std::ostringstream oStr;
00055     boost::archive::text_oarchive oa (oStr);
00056     oa << *this;
00057   }
00058
00059   // ////////////////////////////////////////////////////////////////////
00060   void SegmentCabinKey::serialisationImplementationImport() {
00061     std::istringstream iStr;
00062     boost::archive::text_iarchive ia (iStr);
00063     ia >> *this;
00064   }
00065
```

```
00066   // ////////////////////////////////////////////////////////////////////
00067   template<class Archive>
00068   void SegmentCabinKey::serialize (Archive& ioArchive,
00069                                    const unsigned int iFileVersion) {
00074     ioArchive & _cabinCode;
00075   }
00076
00077   // ////////////////////////////////////////////////////////////////////
00078   // Explicit template instantiation
00079   namespace ba = boost::archive;
00080   template void SegmentCabinKey::
00081   serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
00082   template void SegmentCabinKey::
00083   serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
00084   // ////////////////////////////////////////////////////////////////////
00085
00086 }
```

33.445 stdair/bom/SegmentCabinKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::SegmentCabinKey

  Key of a given segment-cabin, made of a cabin code (only).

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.446 stdair/bom/SegmentCabinKey.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTCABINKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTCABINKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct SegmentCabinKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     SegmentCabinKey();
00035
00036   public:
00040     SegmentCabinKey (const CabinCode_T& iCabinCode);
00041
00045     SegmentCabinKey (const SegmentCabinKey&);
00046
00050     ~SegmentCabinKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const CabinCode_T& getCabinCode() const {
00057       return _cabinCode;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     CabinCode_T _cabinCode;
```

```
00112   };
00113
00114 }
00115 #endif // __STDAIR_BOM_SEGMENTCABINKEY_HPP
```

33.447 stdair/bom/SegmentCabinTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< SegmentCabin * > stdair::SegmentCabinList_T
- typedef std::map< const MapKey_T, SegmentCabin * > stdair::SegmentCabinMap_T

33.448 stdair/bom/SegmentCabinTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class SegmentCabin;
00018
00020   typedef std::list<SegmentCabin*> SegmentCabinList_T;
00021
00023   typedef std::map<const MapKey_T, SegmentCabin*> SegmentCabinMap_T;
00024
00025 }
00026 #endif // __STDAIR_BOM_SEGMENTCABINTYPES_HPP
00027
```

33.449 stdair/bom/SegmentDate.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.450 stdair/bom/SegmentDate.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentDate::SegmentDate()
    : _key (DEFAULT_ORIGIN, DEFAULT_DESTINATION), _parent (NULL),
      _operatingSegmentDate (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDate::SegmentDate (const SegmentDate& iSegmentDate)
    : _key (iSegmentDate._key),
      _parent (NULL),
      _operatingSegmentDate (NULL),
      _boardingDate (iSegmentDate._boardingDate),
      _boardingTime (iSegmentDate._boardingTime),
      _offDate (iSegmentDate._offDate),
      _offTime (iSegmentDate._offTime),
      _elapsedTime (iSegmentDate._elapsedTime),
      _distance (iSegmentDate._distance),
      _routingLegKeyList (iSegmentDate._routingLegKeyList) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDate::SegmentDate (const Key_T& iKey)
    : _key (iKey), _parent (NULL) ,
      _operatingSegmentDate (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDate::~SegmentDate() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SegmentDate::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const Duration_T SegmentDate::getTimeOffset() const {
    // TimeOffset = (OffTime - BoardingTime) + (OffDate - BoardingDate) * 24
    //              - ElapsedTime
    Duration_T oTimeOffset = (_offTime - _boardingTime);
    const DateOffset_T& lDateOffset = getDateOffset();
    const Duration_T lDateOffsetInHours (lDateOffset.days() * 24, 0, 0);
    oTimeOffset += lDateOffsetInHours - _elapsedTime;
    return oTimeOffset;
  }
}
```

33.451 stdair/bom/SegmentDate.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/SegmentDateTypes.hpp>
```

Classes

- class stdair::SegmentDate

  Class representing the actual attributes for an airline segment-date.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< std::string > stdair::RoutingLegKeyList_T

33.452 stdair/bom/SegmentDate.hpp

```
#ifndef __STDAIR_BOM_SEGMENTDATE_HPP
#define __STDAIR_BOM_SEGMENTDATE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
#include <stdair/bom/SegmentDateTypes.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  struct SegmentCabinKey;
  class SegmentCabin;

  // Define the routing leg keys list type.
  typedef std::list<std::string> RoutingLegKeyList_T;

  class SegmentDate : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef SegmentDateKey Key_T;


  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const AirportCode_T& getBoardingPoint() const {
      return _key.getBoardingPoint();
    }

    const AirportCode_T& getOffPoint() const {
      return _key.getOffPoint();
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const Date_T& getBoardingDate() const {
      return _boardingDate;
    }
```

```
00093
00097     const Duration_T& getBoardingTime() const {
00098       return _boardingTime;
00099     }
00100
00104     const Date_T& getOffDate() const {
00105       return _offDate;
00106     }
00107
00111     const Duration_T& getOffTime() const {
00112       return _offTime;
00113     }
00114
00118     const Duration_T& getElapsedTime() const {
00119       return _elapsedTime;
00120     }
00121
00125     const Distance_T& getDistance() const {
00126       return _distance;
00127     }
00128
00132     const DateOffset_T getDateOffset() const {
00133       return _offDate - _boardingDate;
00134     }
00135
00144     const Duration_T getTimeOffset() const;
00145
00149     SegmentDate* getOperatingSegmentDate () const {
00150       return _operatingSegmentDate;
00151     }
00152
00156     const SegmentDateList_T& getMarketingSegmentDateList () const {
00157       return _marketingSegmentDateList;
00158     }
00159
00163     const RoutingLegKeyList_T& getLegKeyList () const {
00164       return _routingLegKeyList;
00165     }
00166
00167   public:
00168     // ///////// Setters //////////
00172     void setBoardingDate (const Date_T& iBoardingDate) {
00173       _boardingDate = iBoardingDate;
00174     }
00175
00179     void setBoardingTime (const Duration_T& iBoardingTime) {
00180       _boardingTime = iBoardingTime;
00181     }
00182
00186     void setOffDate (const Date_T& iOffDate) {
00187       _offDate = iOffDate;
00188     }
00189
00193     void setOffTime (const Duration_T& iOffTime) {
00194       _offTime = iOffTime;
00195     }
00196
00200     void setElapsedTime (const Duration_T& iElapsedTime) {
00201       _elapsedTime = iElapsedTime;
00202     }
00203
00207     void setDistance (const Distance_T& iDistance) {
00208       _distance = iDistance;
00209     }
00210
00214     void addLegKey (const std::string& iLegKey) {
00215       _routingLegKeyList.push_back(iLegKey);
```

```
00216     }
00217
00218   private:
00222     void linkWithOperating (SegmentDate& iSegmentDate) {
00223       _operatingSegmentDate = &iSegmentDate;
00224     }
00225
00226   public:
00227     // /////////// Display support methods /////////
00233     void toStream (std::ostream& ioOut) const {
00234       ioOut << toString();
00235     }
00236
00242     void fromStream (std::istream& ioIn) {
00243     }
00244
00248     std::string toString() const;
00249
00253     const std::string describeKey() const {
00254       return _key.toString();
00255     }
00256
00257
00258   public:
00259     // /////////// (Boost) Serialisation support methods /////////
00263     template<class Archive>
00264     void serialize (Archive& ar, const unsigned int iFileVersion);
00265
00266   private:
00274     void serialisationImplementationExport() const;
00275     void serialisationImplementationImport();
00276
00277
00278   protected:
00279     // ////////// Constructors and destructors /////////
00283     SegmentDate (const Key_T&);
00284
00288     virtual ~SegmentDate();
00289
00290   private:
00294     SegmentDate();
00295
00299     SegmentDate (const SegmentDate&);
00300
00301
00302   protected:
00303     // ////////// Attributes /////////
00307     Key_T _key;
00308
00312     BomAbstract* _parent;
00313
00317     HolderMap_T _holderMap;
00318
00325     SegmentDate* _operatingSegmentDate;
00326
00333     SegmentDateList_T _marketingSegmentDateList;
00334
00338     Date_T _boardingDate;
00339
00343     Duration_T _boardingTime;
00344
00348     Date_T _offDate;
00349
00353     Duration_T _offTime;
00354
00358     Duration_T _elapsedTime;
00359
```

```
    Distance_T _distance;

    RoutingLegKeyList_T _routingLegKeyList;
  };

}
#endif // __STDAIR_BOM_SEGMENTDATE_HPP

```

33.453 stdair/bom/SegmentDateKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentDateKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::SegmentDateKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentDateKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.454 stdair/bom/SegmentDateKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentDateKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentDateKey::SegmentDateKey()
    : _boardingPoint (DEFAULT_ORIGIN), _offPoint (DEFAULT_DESTINATION) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDateKey::SegmentDateKey (const AirportCode_T& iBoardingPoint,
                                  const AirportCode_T& iOffPoint)
    : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDateKey::SegmentDateKey (const SegmentDateKey& iKey)
    : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentDateKey::~SegmentDateKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentDateKey::toStream (std::ostream& ioOut) const {
    ioOut << "SegmentDateKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentDateKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SegmentDateKey::toString() const {
    std::ostringstream oStr;
    oStr << _boardingPoint
         << DEFAULT_KEY_SUB_FLD_DELIMITER << " " << _offPoint;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentDateKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentDateKey::serialisationImplementationImport() {
    std::istringstream iStr;
```

```
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void SegmentDateKey::serialize (Archive& ioArchive,
                                  const unsigned int iFileVersion) {
    ioArchive & _boardingPoint & _offPoint;
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template void SegmentDateKey::serialize<ba::text_oarchive>(ba::text_oarchive&,
                                                             unsigned int);
  template void SegmentDateKey::serialize<ba::text_iarchive>(ba::text_iarchive&,
                                                             unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

33.455 stdair/bom/SegmentDateKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::SegmentDateKey

  Key of a given segment-date, made of an origin and a destination airports.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.456 stdair/bom/SegmentDateKey.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTDATEKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTDATEKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_basic_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00012 namespace boost {
00013   namespace serialization {
00014     class access;
00015   }
00016 }
00017
00018 namespace stdair {
00019
00024   struct SegmentDateKey : public KeyAbstract {
00025     friend class boost::serialization::access;
00026
00027     // /////////// Constructors and destructors ///////////
00028   private:
00032     SegmentDateKey();
00033
00034   public:
00038     SegmentDateKey (const AirportCode_T&, const AirportCode_T&);
00042     SegmentDateKey (const SegmentDateKey&);
00046     ~SegmentDateKey();
00047
00048
00049     // /////////// Getters //////////
00051     const AirportCode_T& getBoardingPoint() const {
00052       return _boardingPoint;
00053     }
00054
00056     const AirportCode_T& getOffPoint() const {
00057       return _offPoint;
00058     }
00059
00060
00061     // /////////// Display support methods /////////
00067     void toStream (std::ostream& ioOut) const;
00068
00074     void fromStream (std::istream& ioIn);
00075
00085     const std::string toString() const;
00086
00087
00088   public:
00089     // /////////// (Boost) Serialisation support methods /////////
00093     template<class Archive>
00094     void serialize (Archive& ar, const unsigned int iFileVersion);
00095
00096   private:
00101     void serialisationImplementationExport() const;
00102     void serialisationImplementationImport();
00103
00104
00105   private:
00106     // ///////////////// Attributes ///////////////
00110     AirportCode_T _boardingPoint;
00111
00115     AirportCode_T _offPoint;
00116   };
```

```
00117
00118 }
00119 #endif // __STDAIR_BOM_SEGMENTDATEKEY_HPP
```

33.457 stdair/bom/SegmentDateTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< SegmentDate * > stdair::SegmentDateList_T
- typedef std::map< const MapKey_T, SegmentDate * > stdair::SegmentDateMap_T

33.458 stdair/bom/SegmentDateTypes.hpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 #ifndef __STDAIR_BOM_SEGMENTDATETYPES_HPP
00003 #define __STDAIR_BOM_SEGMENTDATETYPES_HPP
00004
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <map>
00010 #include <list>
00011 // StdAir
00012 #include <stdair/bom/key_types.hpp>
00013
00014 namespace stdair {
00015
00016   // Forward declarations.
00017   class SegmentDate;
00018
00020   typedef std::list<SegmentDate*> SegmentDateList_T;
00021
00023   typedef std::map<const MapKey_T, SegmentDate*> SegmentDateMap_T;
00024 }
00025
00026 #endif // __STDAIR_BOM_SEGMENTDATETYPES_HPP
00027
```

33.459 stdair/bom/SegmentPeriod.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.460 stdair/bom/SegmentPeriod.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// STDAIR
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/SegmentPeriod.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod::SegmentPeriod (const Key_T& iKey)
    : _key (iKey), _parent (NULL),  _boardingDateOffset (0), _offDateOffset (0) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod::SegmentPeriod (const SegmentPeriod& iSegmentPeriod)
    : _key (iSegmentPeriod.getKey()),
      _parent (NULL),
      _boardingTime (iSegmentPeriod._boardingTime),
      _offTime (iSegmentPeriod._offTime),
      _boardingDateOffset (iSegmentPeriod._boardingDateOffset),
      _offDateOffset (iSegmentPeriod._offDateOffset),
      _elapsedTime (iSegmentPeriod._elapsedTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod::~SegmentPeriod () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SegmentPeriod::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPeriod::
  addCabinBookingClassList (const CabinCode_T& iCabinCode,
                            const ClassList_String_T& iClassCodeList) {
    const bool insert = _cabinBookingClassMap.
      insert (CabinBookingClassMap_T::value_type (iCabinCode,
                                                  iClassCodeList)).second;
    assert (insert == true);
  }

}
```

33.461 stdair/bom/SegmentPeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentPeriodKey.hpp>
#include <stdair/bom/SegmentPeriodTypes.hpp>
```

Classes

- class stdair::SegmentPeriod

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.462 stdair/bom/SegmentPeriod.hpp

```
#ifndef __STDAIR_BOM_SEGMENTPERIOD_HPP
#define __STDAIR_BOM_SEGMENTPERIOD_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentPeriodKey.hpp>
#include <stdair/bom/SegmentPeriodTypes.hpp>

namespace stdair {

  class SegmentPeriod : public BomAbstract {
    template <typename BOM> friend class FacBom;
    template <typename BOM> friend class FacCloneBom;
    friend class FacBomManager;

  public:
    // Type definitions.
    typedef SegmentPeriodKey Key_T;

  public:
    // /////////// Getters /////////////
    const Key_T& getKey() const { return _key; }

    BomAbstract* const getParent() const { return _parent; }

    const AirportCode_T& getBoardingPoint () const {
      return _key.getBoardingPoint();
    }

    const AirportCode_T& getOffPoint () const { return _key.getOffPoint(); }

    const Duration_T& getBoardingTime () const { return _boardingTime; }

    const Duration_T& getOffTime () const { return _offTime; }

    const DateOffset_T& getBoardingDateOffset () const {
      return _boardingDateOffset;
    }

    const DateOffset_T& getOffDateOffset () const { return _offDateOffset; }

    const Duration_T& getElapsedTime() const { return _elapsedTime; }

    const CabinBookingClassMap_T& getCabinBookingClassMap () const {
      return _cabinBookingClassMap;
    }

    const HolderMap_T& getHolderMap() const { return _holderMap; }

  public:
    // ///////// Setters //////////
    void setBoardingTime (const Duration_T& iBoardingTime) {
      _boardingTime = iBoardingTime;
    }

    void setOffTime (const Duration_T& iOffTime) { _offTime = iOffTime; }

    void setBoardingDateOffset (const DateOffset_T& iDateOffset) {
      _boardingDateOffset = iDateOffset;
    }

    void setOffDateOffset (const DateOffset_T& iDateOffset) {
```

```
      _offDateOffset = iDateOffset;
    }

    void setElapsedTime (const Duration_T& iElapsedTime) {
      _elapsedTime = iElapsedTime;
    }

    void addCabinBookingClassList (const CabinCode_T&,
                                   const ClassList_String_T&);

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const { ioOut << toString(); }

    void fromStream (std::istream& ioIn) { }

    std::string toString() const;

    const std::string describeKey() const { return _key.toString(); }

  protected:
    // ////////// Constructors and destructors /////////
    SegmentPeriod (const Key_T&);
    virtual ~SegmentPeriod();

  private:
    SegmentPeriod();
    SegmentPeriod (const SegmentPeriod&);

  protected:
    // Attributes
    Key_T _key;
    BomAbstract* _parent;
    Duration_T _boardingTime;
    Duration_T _offTime;
    DateOffset_T _boardingDateOffset;
    DateOffset_T _offDateOffset;
    Duration_T _elapsedTime;
    CabinBookingClassMap_T _cabinBookingClassMap;
    HolderMap_T _holderMap;
  };

}
#endif // __STDAIR_BOM_SEGMENTPERIOD_HPP
```

33.463 stdair/bom/SegmentPeriodKey.cpp File Reference

```
#include <sstream>
#include <stdair/bom/SegmentPeriodKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.464 stdair/bom/SegmentPeriodKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
// StdAir
#include <stdair/bom/SegmentPeriodKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriodKey::SegmentPeriodKey (const AirportCode_T& iBoardingPoint,
                                      const AirportCode_T& iOffPoint)
    : _boardingPoint (iBoardingPoint), _offPoint (iOffPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriodKey::SegmentPeriodKey (const SegmentPeriodKey& iKey)
    : _boardingPoint (iKey._boardingPoint), _offPoint (iKey._offPoint) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriodKey::~SegmentPeriodKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPeriodKey::toStream (std::ostream& ioOut) const {
    ioOut << "SegmentPeriodKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentPeriodKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SegmentPeriodKey::toString() const {
    std::ostringstream oStr;
    oStr << _boardingPoint << "-" << _offPoint;
    return oStr.str();
  }

}
```

33.465 stdair/bom/SegmentPeriodKey.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::SegmentPeriodKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.466 stdair/bom/SegmentPeriodKey.hpp

```
#ifndef __STDAIR_BOM_SEGMENTPERIODKEY_HPP
#define __STDAIR_BOM_SEGMENTPERIODKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>

namespace stdair {

  struct SegmentPeriodKey : public KeyAbstract {

  private:
    // /////////// Default constructor //////////
    SegmentPeriodKey () { };
  public:
    // /////////// Construction ///////////
    SegmentPeriodKey (const AirportCode_T&, const AirportCode_T&);
    SegmentPeriodKey (const SegmentPeriodKey&);
    ~SegmentPeriodKey ();

    // /////////// Getters //////////
    const AirportCode_T& getBoardingPoint() const {
      return _boardingPoint;
    }

    const AirportCode_T& getOffPoint() const {
      return _offPoint;
    }

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

  private:
    // Attributes
    AirportCode_T _boardingPoint;

    AirportCode_T _offPoint;
  };

}
#endif // __STDAIR_BOM_SEGMENTPERIODKEY_HPP
```

33.467 stdair/bom/SegmentPeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< SegmentPeriod * > stdair::SegmentPeriodList_T
- typedef std::map< const MapKey_T, SegmentPeriod * > stdair::SegmentPeriodMap_T
- typedef std::pair< MapKey_T, SegmentPeriod * > stdair::SegmentPeriodWithKey_T
- typedef std::list< SegmentPeriodWithKey_T > stdair::SegmentPeriodDetailedList_T

33.468 stdair/bom/SegmentPeriodTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_SEGMENTPERIODTYPES_HPP
#define __STDAIR_BOM_SEGMENTPERIODTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class SegmentPeriod;

  typedef std::list<SegmentPeriod*> SegmentPeriodList_T;

  typedef std::map<const MapKey_T, SegmentPeriod*> SegmentPeriodMap_T;

  typedef std::pair<MapKey_T, SegmentPeriod*> SegmentPeriodWithKey_T;
  typedef std::list<SegmentPeriodWithKey_T> SegmentPeriodDetailedList_T;
}
#endif // __STDAIR_BOM_SEGMENTPERIODTYPES_HPP

```

33.469 stdair/bom/SegmentSnapshotTable.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/multi_array.hpp>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.470 stdair/bom/SegmentSnapshotTable.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#include <boost/multi_array.hpp>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentSnapshotTable.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTable::SegmentSnapshotTable()
    : _key (DEFAULT_TABLE_ID), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTable::SegmentSnapshotTable (const SegmentSnapshotTable&)
    : _key (DEFAULT_TABLE_ID), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTable::
  SegmentSnapshotTable (const Key_T& iKey) : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTable::~SegmentSnapshotTable() {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string SegmentSnapshotTable::toString() const {
    std::ostringstream oStr;
    oStr << describeKey();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTable::
  initSnapshotBlocks (const SegmentCabinIndexMap_T& iSegmentCabinIndexMap,
                      const ClassIndexMap_T& iClassIndexMap) {
    _segmentCabinIndexMap = iSegmentCabinIndexMap;
    _classIndexMap = iClassIndexMap;

    unsigned int lNumberOfSegmentCabins = _segmentCabinIndexMap.size();
    unsigned int lNumberOfClasses = _classIndexMap.size();

    // Initialise the snapshot blocks
    // Normally, the block includes snapshots from DTD MAX to DTD 0, thus
    // DEFAULT_MAX_DTD + 1 values. However, we would like to add the day
    // before DTD MAX (this value will be initialised to zero).
    _bookingSnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _cancellationSnapshotBlock.
```

```
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _productOrientedNetBookingSnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _priceOrientedNetBookingSnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _productOrientedGrossBookingSnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _priceOrientedGrossBookingSnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);
    _availabilitySnapshotBlock.
      resize (boost::extents[lNumberOfSegmentCabins*lNumberOfClasses]
              [DEFAULT_MAX_DTD + 2]);

  }

  // ////////////////////////////////////////////////////////////////////
  const ClassIndex_T& SegmentSnapshotTable::
  getClassIndex (const MapKey_T& iKey) const {
    ClassIndexMap_T::const_iterator itVTIdx =
      _classIndexMap.find (iKey);
    assert (itVTIdx != _classIndexMap.end());
    return itVTIdx->second;
  }

  // ////////////////////////////////////////////////////////////////////
  const SegmentDataID_T& SegmentSnapshotTable::
  getSegmentDataID (const SegmentCabin& iSegmentCabin) const {
    SegmentCabinIndexMap_T::const_iterator itSCIdx =
      _segmentCabinIndexMap.find (&iSegmentCabin);
    assert (itSCIdx != _segmentCabinIndexMap.end());
    return itSCIdx->second;
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                              const SegmentDataID_T iSCIdxEnd,
                                              const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangeBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
```

```
                                        const SegmentDataID_T iSCIdxEnd,
                                        const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                              const SegmentDataID_T iSCIdxEnd,
                                              const DTD_T iDTDBegin,
                                              const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _bookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                              const SegmentDataID_T iSCIdxEnd,
                                              const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangeCancellationSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                        const SegmentDataID_T iSCIdxEnd,
                                        const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangeCancellationSnapshotView(const SegmentDataID_T iSCIdxBeg
```

```
in,
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTDBegin,
                                                const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _cancellationSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _productOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDProductOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                          const SegmentDataID_T iSCIdxEnd,
                                          const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTDBegin,
                                                const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedNetBookingSnapshotBlock [ boost::indices[
```

```
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }


  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                                              const SegmentDataID_T iSCIdxEnd,
                                                              const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _priceOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDPriceOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                                         const SegmentDataID_T iSCIdxEnd,
                                                         const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
                                                              const SegmentDataID_T iSCIdxEnd,
                                                              const DTD_T iDTDBegin,
                                                              const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedNetBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

    // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const SegmentDataID_T iSCIdxBegin,
```

```
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _productOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegi
n, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDProductOrientedGrossBookingSnapshotView (const
SegmentDataID_T iSCIdxBegin,
                                        const SegmentDataID_T iSCIdxEnd,
                                        const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView (const
SegmentDataID_T iSCIdxBegin,
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTDBegin,
                                                const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _productOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegi
n, iDTDEnd + 1)] ];
  }


  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView (const
SegmentDataID_T iSCIdxBegin,
                                                const SegmentDataID_T iSCIdxEnd,
                                                const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[
SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _priceOrientedGrossBookingSnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDAvailabilitySnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTD) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  ConstSegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getConstSegmentCabinDTDRangeAvailabilitySnapshotView
  (const SegmentDataID_T iSCIdxBegin, const SegmentDataID_T iSCIdxEnd,
   const DTD_T iDTDBegin, const DTD_T iDTDEnd) const {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd +1) * lNbOfClasses;

    return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
```

```
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T iSCIdxBegin,
                                               const SegmentDataID_T iSCIdxEnd,
                                               const DTD_T iDTD) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][iDTD] ];
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentCabinDTDRangeSnapshotView_T SegmentSnapshotTable::
  getSegmentCabinDTDRangeAvailabilitySnapshotView(const SegmentDataID_T iSCIdxBegin,
                                                  const SegmentDataID_T iSCIdxEnd,
                                                  const DTD_T iDTDBegin,
                                                  const DTD_T iDTDEnd) {
    const unsigned int lNbOfClasses = _classIndexMap.size();
    const unsigned int lClassIdxBegin = iSCIdxBegin * lNbOfClasses;
    const unsigned int lClassIdxEnd = (iSCIdxEnd + 1) * lNbOfClasses;

    return _availabilitySnapshotBlock [ boost::indices[SnapshotBlockRange_T(lClassIdxBegin, lClassIdxEnd)][SnapshotBlockRange_T(iDTDBegin, iDTDEnd + 1)] ];
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTable::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTable::serialisationImplementationImport() {
    std::istringstream iStr;
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void SegmentSnapshotTable::serialize (Archive& ioArchive,
                                        const unsigned int iFileVersion) {
    ioArchive & _key;
  }

}
```

33.471 stdair/bom/SegmentSnapshotTable.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/SegmentSnapshotTableKey.hpp>
#include <stdair/bom/SegmentSnapshotTableTypes.hpp>
```

Classes

- class stdair::SegmentSnapshotTable

  Class representing the actual attributes for an airline segment data tables.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.472 stdair/bom/SegmentSnapshotTable.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTSNAPSHOTTABLE_HPP
00002 #define __STDAIR_BOM_SEGMENTSNAPSHOTTABLE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/bom/BomAbstract.hpp>
00013 #include <stdair/bom/SegmentSnapshotTableKey.hpp>
00014 #include <stdair/bom/SegmentSnapshotTableTypes.hpp>
00015
00017 namespace boost {
00018   namespace serialization {
00019     class access;
00020   }
00021 }
00022
00023 namespace stdair {
00024   // Forward declarations
00025   class SegmentCabin;
00026
00031   class SegmentSnapshotTable : public BomAbstract {
00032     template <typename BOM> friend class FacBom;
00033     friend class FacBomManager;
00034     friend class boost::serialization::access;
00035
00036   public:
00037     // ////////// Type definitions ////////////
00041     typedef SegmentSnapshotTableKey Key_T;
00042
00043
00044   public:
00045     // /////////// Getters ///////////////
00047     const Key_T& getKey() const {
00048       return _key;
00049     }
00050
00052     BomAbstract* const getParent() const {
00053       return _parent;
00054     }
00055
00057     const TableID_T& getTableID() const {
00058       return _key.getTableID();
00059     }
00060
00064     const HolderMap_T& getHolderMap() const {
00065       return _holderMap;
00066     }
00067
00069     const SegmentCabinIndexMap_T& getSegmentCabinIndexMap() const {
00070       return _segmentCabinIndexMap;
00071     }
00072
00074     const ClassIndexMap_T& getClassIndexMap() const {
00075       return _classIndexMap;
00076     }
00077
00079     const ClassIndex_T& getClassIndex (const MapKey_T&) const;
00080
00082     const SegmentDataID_T& getSegmentDataID (const SegmentCabin&) const;
00083
```

```
    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T,
                                                const SegmentDataID_T,
                                                const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T,
                                                     const SegmentDataID_T,
                                                     const DTD_T,
                                                     const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDBookingSnapshotView (const SegmentDataID_T,
                                           const SegmentDataID_T, const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangeBookingSnapshotView (const SegmentDataID_T,
                                                const SegmentDataID_T,
                                                const DTD_T, const DTD_T);

    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T,
                                                const SegmentDataID_T,
                                                const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T,
                                                          const SegmentDataID_T,
                                                          const DTD_T,
                                                          const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDCancellationSnapshotView (const SegmentDataID_T,
                                                const SegmentDataID_T,
                                                const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangeCancellationSnapshotView (const SegmentDataID_T,
                                                     const SegmentDataID_T,
                                                     const DTD_T, const DTD_T);

    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDProductOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDProductOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangeProductOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T);

    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDPriceOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const;

```

```
    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDPriceOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangePriceOrientedNetBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T);


    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDProductOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDProductOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangeProductOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T);

    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDPriceOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDPriceOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangePriceOrientedGrossBookingSnapshotView
    (const SegmentDataID_T, const SegmentDataID_T, const DTD_T, const DTD_T);


    ConstSegmentCabinDTDSnapshotView_T
    getConstSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T,
                                                      const SegmentDataID_T,
                                                      const DTD_T) const;

    ConstSegmentCabinDTDRangeSnapshotView_T
    getConstSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T,
                                                           const SegmentDataID_T,
                                                           const DTD_T,
                                                           const DTD_T) const;

    SegmentCabinDTDSnapshotView_T
    getSegmentCabinDTDAvailabilitySnapshotView (const SegmentDataID_T,
                                                 const SegmentDataID_T,
                                                 const DTD_T);

    SegmentCabinDTDRangeSnapshotView_T
    getSegmentCabinDTDRangeAvailabilitySnapshotView (const SegmentDataID_T,
                                                      const SegmentDataID_T,
                                                      const DTD_T, const DTD_T);


  public:
```

```
00270     // ///////////// Setters /////////////
00273     void initSnapshotBlocks (const SegmentCabinIndexMap_T&,
00274                              const ClassIndexMap_T&);
00275
00276   public:
00277     // /////////// Display support methods /////////
00283     void toStream (std::ostream& ioOut) const {
00284       ioOut << toString();
00285     }
00286
00292     void fromStream (std::istream& ioIn) {
00293     }
00294
00298     std::string toString() const;
00299
00303     const std::string describeKey() const {
00304       return _key.toString();
00305     }
00306
00307
00308   public:
00309     // /////////// (Boost) Serialisation support methods /////////
00313     template<class Archive>
00314     void serialize (Archive& ar, const unsigned int iFileVersion);
00315
00316   private:
00321     void serialisationImplementationExport() const;
00322     void serialisationImplementationImport();
00323
00324
00325   protected:
00326     // ////////// Constructors and destructors /////////
00330     SegmentSnapshotTable (const Key_T&);
00331
00335     virtual ~SegmentSnapshotTable();
00336
00337   private:
00341     SegmentSnapshotTable();
00342
00346     SegmentSnapshotTable (const SegmentSnapshotTable&);
00347
00348
00349   protected:
00350     // ////////// Attributes /////////
00352     Key_T _key;
00353
00355     BomAbstract* _parent;
00356
00358     HolderMap_T _holderMap;
00359
00361     SegmentCabinIndexMap_T _segmentCabinIndexMap;
00362
00365     ClassIndexMap_T _classIndexMap;
00366
00368     SnapshotBlock_T _bookingSnapshotBlock;
00369
00371     SnapshotBlock_T _cancellationSnapshotBlock;
00372
00374     SnapshotBlock_T _productOrientedNetBookingSnapshotBlock;
00375
00377     SnapshotBlock_T _priceOrientedNetBookingSnapshotBlock;
00378
00380     SnapshotBlock_T _productOrientedGrossBookingSnapshotBlock;
00381
00383     SnapshotBlock_T _priceOrientedGrossBookingSnapshotBlock;
00384
00386     SnapshotBlock_T _availabilitySnapshotBlock;
```

```
  };

}
#endif // __STDAIR_BOM_SEGMENTSNAPSHOTTABLE_HPP

```

33.473 stdair/bom/SegmentSnapshotTableKey.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentSnapshotTableKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template void stdair::SegmentSnapshotTableKey::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentSnapshotTableKey::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.474 stdair/bom/SegmentSnapshotTableKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost.Serialization
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/access.hpp>
// StdAir
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/basic/BasConst_BomDisplay.hpp>
#include <stdair/bom/SegmentSnapshotTableKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTableKey::SegmentSnapshotTableKey()
    : _tableID (DEFAULT_TABLE_ID) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTableKey::
  SegmentSnapshotTableKey (const TableID_T& iTableID)
    : _tableID (iTableID) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTableKey::SegmentSnapshotTableKey (const
SegmentSnapshotTableKey& iKey)
    : _tableID (iKey._tableID) {
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentSnapshotTableKey::~SegmentSnapshotTableKey() {
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTableKey::toStream (std::ostream& ioOut) const {
    ioOut << "SegmentSnapshotTableKey: " << toString();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTableKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SegmentSnapshotTableKey::toString() const {
    std::ostringstream oStr;
    oStr << _tableID;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTableKey::serialisationImplementationExport() const {
    std::ostringstream oStr;
    boost::archive::text_oarchive oa (oStr);
    oa << *this;
  }

  // ////////////////////////////////////////////////////////////////////
  void SegmentSnapshotTableKey::serialisationImplementationImport() {
    std::istringstream iStr;
```

```
    boost::archive::text_iarchive ia (iStr);
    ia >> *this;
  }

  // ////////////////////////////////////////////////////////////////////
  template<class Archive>
  void SegmentSnapshotTableKey::serialize (Archive& ioArchive,
                                           const unsigned int iFileVersion) {
    ioArchive & _tableID;
  }

  // ////////////////////////////////////////////////////////////////////
  // Explicit template instantiation
  namespace ba = boost::archive;
  template void SegmentSnapshotTableKey::
  serialize<ba::text_oarchive> (ba::text_oarchive&, unsigned int);
  template void SegmentSnapshotTableKey::
  serialize<ba::text_iarchive> (ba::text_iarchive&, unsigned int);
  // ////////////////////////////////////////////////////////////////////

}
```

33.475 stdair/bom/SegmentSnapshotTableKey.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::SegmentSnapshotTableKey

  Key of a given guillotine block, made of a guillotine number.

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.476 stdair/bom/SegmentSnapshotTableKey.hpp

```
00001 #ifndef __STDAIR_BOM_SEGMENTSNAPSHOTTABLEKEY_HPP
00002 #define __STDAIR_BOM_SEGMENTSNAPSHOTTABLEKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_basic_types.hpp>
00012 #include <stdair/bom/KeyAbstract.hpp>
00013
00015 namespace boost {
00016   namespace serialization {
00017     class access;
00018   }
00019 }
00020
00021 namespace stdair {
00022
00026   struct SegmentSnapshotTableKey : public KeyAbstract {
00027     friend class boost::serialization::access;
00028
00029     // /////////// Constructors and destructors ///////////
00030   private:
00034     SegmentSnapshotTableKey();
00035
00036   public:
00040     SegmentSnapshotTableKey (const TableID_T&);
00041
00045     SegmentSnapshotTableKey (const SegmentSnapshotTableKey&);
00046
00050     ~SegmentSnapshotTableKey();
00051
00052
00053   public:
00054     // /////////// Getters //////////
00056     const TableID_T& getTableID() const {
00057       return _tableID;
00058     }
00059
00060
00061   public:
00062     // /////////// Display support methods /////////
00068     void toStream (std::ostream& ioOut) const;
00069
00075     void fromStream (std::istream& ioIn);
00076
00086     const std::string toString() const;
00087
00088
00089   public:
00090     // /////////// (Boost) Serialisation support methods /////////
00094     template<class Archive>
00095     void serialize (Archive& ar, const unsigned int iFileVersion);
00096
00097   private:
00102     void serialisationImplementationExport() const;
00103     void serialisationImplementationImport();
00104
00105
00106   private:
00107     // ///////////////// Attributes ///////////////
00111     TableID_T _tableID;
```

```
00112
00113   };
00114
00115 }
00116 #endif // __STDAIR_BOM_SEGMENTSNAPSHOTTABLEKEY_HPP
```

33.477 stdair/bom/SegmentSnapshotTableTypes.hpp File Reference

```
#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< SegmentSnapshotTable * > stdair::SegmentSnapshotTableList_T
- typedef std::map< const MapKey_T, SegmentSnapshotTable * > stdair::SegmentSnapshotTableMap_T
- typedef std::map< const SegmentCabin *, SegmentDataID_T > stdair::SegmentCabinIndexMap_T
- typedef std::map< const MapKey_T, ClassIndex_T > stdair::ClassIndexMap_T

33.478 stdair/bom/SegmentSnapshotTableTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_SEGMENTSNAPSHOTTABLETYPES_HPP
#define __STDAIR_BOM_SEGMENTSNAPSHOTTABLETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// BOOST
#include <boost/multi_array.hpp>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations
  class SegmentSnapshotTable;
  class SegmentCabin;

  // ////////////////////////// Type definitions //////////////////////////
  typedef std::list<SegmentSnapshotTable*> SegmentSnapshotTableList_T;

  typedef std::map<const MapKey_T, SegmentSnapshotTable*>
SegmentSnapshotTableMap_T;

  typedef std::map<const SegmentCabin*, SegmentDataID_T> SegmentCabinIndexMap_T;

  typedef std::map<const MapKey_T, ClassIndex_T> ClassIndexMap_T;

}
#endif // __STDAIR_BOM_SEGMENTSNAPSHOTTABLETYPES_HPP
```

33.479 stdair/bom/SimpleNestingStructure.cpp File Reference

```
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.480 stdair/bom/SimpleNestingStructure.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <sstream>
#include <cassert>
#include <iomanip>
#include <iostream>
//STDAIR
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SimpleNestingStructure::SimpleNestingStructure () :
    _key (DEFAULT_NESTING_STRUCTURE_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SimpleNestingStructure::
  SimpleNestingStructure (const SimpleNestingStructure& iSimpleNestingStructure)
  : _key (DEFAULT_NESTING_STRUCTURE_CODE), _parent (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SimpleNestingStructure::SimpleNestingStructure (const Key_T& iKey)
  : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  SimpleNestingStructure::~SimpleNestingStructure() {
  }

  // ////////////////////////////////////////////////////////////////////
  // const bool SimpleNestingStructure::insertBookingClassList(const Yield_T& iYield,
  //                                   const BookingClassList_T& iBookingClassList) {
  //   const bool isBookinClassListEmpty = iBookingClassList.empty();
  //   if (isBookinClassListEmpty == true) {
  //     std::ostringstream ostr;
  //     ostr << "The booking class list is empty and it should not be. "
  //       "No insertion done in the nesting structure (";
  //     toStream(ostr);
  //     ostr << ").";
  //     STDAIR_LOG_DEBUG(ostr.str());
  //     throw BookingClassListEmptyInNestingStructException(ostr.str());
  //   }
  //   assert(isBookinClassListEmpty == false);
  //   NestingNodeMap_T::iterator itNestingNode = _nestingNodeMap.find (iYield);
  //   bool insertionSucceeded = false;
  //   // Search a node with the same yield and add the
  //   // booking classes to the booking class list of the node.
  //   // If there is not a node with the same yield, create it.
  //   if (itNestingNode == _nestingNodeMap.end()) {
```

```
//      NestingNode_T lNode (iYield, iBookingClassList);
//      insertionSucceeded = _nestingNodeMap.insert(lNode).second;
//    } else {
//      NestingNode_T& lNode = *itNestingNode;
//      const Yield_T& lYield = lNode.first;
//      assert(lYield == iYield);
//      BookingClassList_T& lBCList = lNode.second;
//      for (BookingClassList_T::const_iterator itBC = iBookingClassList.begin();
//          itBC != iBookingClassList.end(); ++itBC) {
//        BookingClass* lBC_ptr = *itBC;
//        assert(lBC_ptr != NULL);
//        lBCList.push_back(lBC_ptr);
//      }
//      insertionSucceeded = true;
//    }

//    return insertionSucceeded;
// }

// ////////////////////////////////////////////////////////////////////
// const bool SimpleNestingStructure::
// alreadyInNestingStructure(const ClassCode_T& iClassCode) const {
//    bool isAlreadyInTheMap = false;
//    NestingNodeMap_T::const_iterator itMap = _nestingNodeMap.begin();
//    for(; itMap != _nestingNodeMap.end(); ++itMap) {
//      const NestingNode_T& lNestingNode = *itMap;
//      const BookingClassList_T& lBCList = lNestingNode.second;
//      BookingClassList_T::const_iterator itBC = lBCList.begin();
//      for (;itBC != lBCList.end(); ++itBC) {
//        BookingClass* lBC_ptr = *itBC;
//        assert(lBC_ptr != NULL);
//        const BookingClassKey& lBookingClassKey = lBC_ptr->getKey();
//        const ClassCode_T& lClassCode = lBookingClassKey.getClassCode();
//        if (lClassCode == iClassCode) {
//          isAlreadyInTheMap = true;
//          return isAlreadyInTheMap;
//        }
//      }
//    }
//    return isAlreadyInTheMap;
// }

// ////////////////////////////////////////////////////////////////////
std::string SimpleNestingStructure::toString () const {
  std::ostringstream oStr;
  oStr << describeKey();

  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
const NestingNodeList_T& SimpleNestingStructure::getNestingNodeList() const {
  return BomManager::getList<NestingNode> (*this);
}

}
```

33.481 stdair/bom/SimpleNestingStructure.hpp File Reference

```
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/bom/SimpleNestingStructureTypes.hpp>
#include <stdair/bom/NestingStructureKey.hpp>
```

Classes

- class stdair::SimpleNestingStructure

Namespaces

- namespace boost

  Forward declarations.

- namespace boost::serialization
- namespace stdair

  Handle on the StdAir library context.

33.482 stdair/bom/SimpleNestingStructure.hpp

```
#ifndef __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP
#define __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/NestingNodeTypes.hpp>
#include <stdair/bom/SimpleNestingStructureTypes.hpp>
#include <stdair/bom/NestingStructureKey.hpp>

namespace boost {
  namespace serialization {
    class access;
  }
}

namespace stdair {

  class SimpleNestingStructure : public BomAbstract {
    template <typename BOM> friend class FacBom;
    friend class FacBomManager;
    friend class boost::serialization::access;

  public:
    // ////////// Type definitions ////////////
    typedef NestingStructureKey Key_T;

  public:
    // ////////////////// Getters ////////////////////////
    const Key_T& getKey() const {
      return _key;
    }

    BomAbstract* const getParent() const {
      return _parent;
    }

    const HolderMap_T& getHolderMap() const {
      return _holderMap;
    }

    const NestingNodeList_T& getNestingNodeList() const;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const {
      ioOut << toString();
    }

    void fromStream (std::istream& ioIn) {
    }

    std::string toString() const;

    const std::string describeKey() const {
      return _key.toString();
    }


  public:
    // /////////// (Boost) Serialisation support methods /////////
    template<class Archive>
```

```
    void serialize (Archive& ar, const unsigned int iFileVersion);

  private:
    void serialisationImplementationExport() const;
    void serialisationImplementationImport();


  public:
    // /////////// Constructors and destructor. ////////////
    SimpleNestingStructure (const Key_T&);

    virtual ~SimpleNestingStructure();

  private:
    SimpleNestingStructure();

    SimpleNestingStructure (const SimpleNestingStructure&);

  private:
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;
  };
}
#endif // __STDAIR_BOM_SIMPLENESTINGSTRUCTURE_HPP
```

33.483 stdair/bom/SimpleNestingStructureTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< SimpleNestingStructure * > stdair::SimpleNestingStructureList_T
- typedef std::map< const MapKey_T, SimpleNestingStructure * > stdair::SimpleNestingStructureMap_T

33.484 stdair/bom/SimpleNestingStructureTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP
#define __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class SimpleNestingStructure;

  typedef std::list<SimpleNestingStructure*> SimpleNestingStructureList_T;

  typedef std::map<const MapKey_T, SimpleNestingStructure*>
SimpleNestingStructureMap_T;

}
#endif // __STDAIR_BOM_SIMPLENESTINGSTRUCTURETYPES_HPP
```

33.485 stdair/bom/SnapshotStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/SnapshotStruct.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.486 stdair/bom/SnapshotStruct.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/bom/SnapshotStruct.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  SnapshotStruct::SnapshotStruct() {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  SnapshotStruct::
  SnapshotStruct (const SnapshotStruct& iSnapshot)
    : _airlineCode (iSnapshot._airlineCode),
      _snapshotTime (iSnapshot._snapshotTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  SnapshotStruct::
  SnapshotStruct (const AirlineCode_T& iAirlineCode,
                  const DateTime_T& iSnapshotTime)
    : _airlineCode (iAirlineCode), _snapshotTime (iSnapshotTime) {
  }

  // ////////////////////////////////////////////////////////////////////
  SnapshotStruct::~SnapshotStruct() {
  }

  // ////////////////////////////////////////////////////////////////////
  void SnapshotStruct::toStream (std::ostream& ioOut) const {
    ioOut << describe();
  }

  // ////////////////////////////////////////////////////////////////////
  void SnapshotStruct::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string SnapshotStruct::describe() const {
    std::ostringstream oStr;
    oStr << _airlineCode << ", " << _snapshotTime;
    return oStr.str();
  }

}
```

33.487 stdair/bom/SnapshotStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/SnapshotTypes.hpp>
```

Classes

- struct stdair::SnapshotStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.488 stdair/bom/SnapshotStruct.hpp

```
00001 #ifndef __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
00002 #define __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <string>
00010 // StdAir
00011 #include <stdair/stdair_inventory_types.hpp>
00012 #include <stdair/stdair_demand_types.hpp>
00013 #include <stdair/basic/StructAbstract.hpp>
00014 #include <stdair/bom/SnapshotTypes.hpp>
00015
00016 namespace stdair {
00017
00019   struct SnapshotStruct : public StructAbstract {
00020   public:
00021     // /////////////// Getters /////////////////
00023     const AirlineCode_T& getAirlineCode() const {
00024       return _airlineCode;
00025     }
00026
00028     const DateTime_T& getSnapshotTime() const {
00029       return _snapshotTime;
00030     }
00031
00032     // /////////// Display support method /////////////
00035     void toStream (std::ostream& ioOut) const;
00036
00039     void fromStream (std::istream& ioIn);
00040
00042     const std::string describe() const;
00043
00044
00045     // /////////////// Constructors and Destructors /////////////////
00046   public:
00048     SnapshotStruct (const AirlineCode_T&, const DateTime_T&);
00049
00051     SnapshotStruct (const SnapshotStruct&);
00052
00053   private:
00056     SnapshotStruct ();
00057
00058   public:
00060     ~SnapshotStruct();
00061
00062
00063   private:
00064     // /////////////// Attributes /////////////////
00066     const AirlineCode_T _airlineCode;
00067
00069     const DateTime_T _snapshotTime;
00070   };
00071
00072 }
00073 #endif // __STDAIR_BOM_SNAPSHOTSTRUCT_HPP
```

33.489 stdair/bom/SnapshotTypes.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< SnapshotStruct > stdair::SnapshotPtr_T

33.490 stdair/bom/SnapshotTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_SNAPSHOTTYPES_HPP
#define __STDAIR_BOM_SNAPSHOTTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// Boost
#include <boost/shared_ptr.hpp>

namespace stdair {

  // Forward declarations
  struct SnapshotStruct;

  // ////////////// Type definitions ///////////////
  typedef boost::shared_ptr<SnapshotStruct> SnapshotPtr_T;

}
#endif // __STDAIR_BOM_SNAPSHOTTYPES_HPP

```

33.491 stdair/bom/TimePeriod.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/TimePeriod.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.492 stdair/bom/TimePeriod.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_General.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/TimePeriod.hpp>
00011
00012 namespace stdair {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   TimePeriod::TimePeriod()
00016     : _key (DEFAULT_EPSILON_DURATION, DEFAULT_EPSILON_DURATION),
00017       _parent (NULL)  {
00018     // That constructor is used by the serialisation process
00019   }
00020
00021   // ////////////////////////////////////////////////////////////////////
00022   TimePeriod::TimePeriod (const TimePeriod& iTimePeriod)
00023     : _key (iTimePeriod.getKey()), _parent (NULL)  {
00024   }
00025
00026   // ////////////////////////////////////////////////////////////////////
00027   TimePeriod::TimePeriod (const Key_T& iKey)
00028     : _key (iKey), _parent (NULL)  {
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   TimePeriod::~TimePeriod () {
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   std::string TimePeriod::toString() const {
00037     std::ostringstream oStr;
00038     oStr << describeKey();
00039     return oStr.str();
00040   }
00041
00042   // ////////////////////////////////////////////////////////////////////
00043   bool TimePeriod::
00044   isDepartureTimeValid (const Time_T& iFlightTime) const {
00045
00046     const Time_T& lTimeRangeStart = getTimeRangeStart();
00047     const Time_T& lTimeRangeEnd = getTimeRangeEnd();
00048
00049     // Check if the departure time is within the time range.
00050     if (lTimeRangeStart >= iFlightTime) {
00051       // DEBUG
00052       STDAIR_LOG_DEBUG ("Time range begin: " << lTimeRangeStart << ", "
00053                         << "time: " << iFlightTime);
00054       return false;
00055     }
00056     if (lTimeRangeEnd <= iFlightTime) {
00057       // DEBUG
00058       STDAIR_LOG_DEBUG ("Time range end: " << lTimeRangeEnd << ", "
00059                         << "time: " << iFlightTime);
00060       return false;
00061     }
00062
00063     return true;
00064   }
00065
```

```
00066 }
00067
```

33.493 stdair/bom/TimePeriod.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
#include <stdair/bom/TimePeriodTypes.hpp>
```

Classes

- class stdair::TimePeriod

  Class representing the actual attributes for a fare time-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.494 stdair/bom/TimePeriod.hpp

```
00001 #ifndef __STDAIR_BOM_FARETIMEPERIOD_HPP
00002 #define __STDAIR_BOM_FARETIMEPERIOD_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STDAIR
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/TimePeriodKey.hpp>
00010 #include <stdair/bom/TimePeriodTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00018   class TimePeriod : public BomAbstract {
00019     template <typename BOM> friend class FacBom;
00020     template <typename BOM> friend class FacCloneBom;
00021     friend class FacBomManager;
00022
00023   public:
00024     // ////////////// Type definitions //////////////
00028     typedef TimePeriodKey Key_T;
00029
00030   public:
00031     // /////////// Display support methods /////////
00032     // /////////// Display support methods /////////
00038     void toStream (std::ostream& ioOut) const {
00039       ioOut << toString();
00040     }
00041
00047     void fromStream (std::istream& ioIn) {
00048     }
00049
00053     std::string toString() const;
00054
00058     const std::string describeKey() const {
00059       return _key.toString();
00060     }
00061
00062   public:
00063     // ////////// Getters ////////////
00067     const Key_T& getKey() const {
00068       return _key;
00069     }
00070
00074     BomAbstract* const getParent() const {
00075       return _parent;
00076     }
00077
00081     const HolderMap_T& getHolderMap() const {
00082       return _holderMap;
00083     }
00084
00088     const Time_T& getTimeRangeStart() const {
00089       return _key.getTimeRangeStart();
00090     }
00091
00095     const Time_T& getTimeRangeEnd() const {
00096       return _key.getTimeRangeEnd();
00097     }
00098
00099   public:
00100     // ////////////// Business methods ///////////////
00105     bool isDepartureTimeValid (const Time_T&) const;
00106
```

```
  protected:
    // ////////// Constructors and destructors /////////
    TimePeriod (const Key_T&);
    virtual ~TimePeriod();

  private:
    TimePeriod();
    TimePeriod (const TimePeriod&);

  protected:
    // ////////////// Attributes //////////////
    Key_T _key;

    BomAbstract* _parent;

    HolderMap_T _holderMap;

  };

}
#endif // __STDAIR_BOM_FARETIMEPERIOD_HPP
```

33.495 stdair/bom/TimePeriodKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/TimePeriodKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.496 stdair/bom/TimePeriodKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// STDAIR
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/TimePeriodKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  TimePeriodKey::TimePeriodKey ()
    : _timeRangeStart (DEFAULT_EPSILON_DURATION),
      _timeRangeEnd (DEFAULT_EPSILON_DURATION) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  TimePeriodKey::TimePeriodKey (const Time_T& iTimeRangeStart,
                                const Time_T& iTimeRangeEnd)
    : _timeRangeStart(iTimeRangeStart),
      _timeRangeEnd(iTimeRangeEnd) {
  }

  // ////////////////////////////////////////////////////////////////////
  TimePeriodKey::TimePeriodKey (const TimePeriodKey& iKey)
    : _timeRangeStart(iKey.getTimeRangeStart()),
      _timeRangeEnd(iKey.getTimeRangeEnd())  {
  }

  // ////////////////////////////////////////////////////////////////////
  TimePeriodKey::~TimePeriodKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void TimePeriodKey::toStream (std::ostream& ioOut) const {
    ioOut << "TimePeriodKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void TimePeriodKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string TimePeriodKey::toString() const {
    std::ostringstream oStr;
    oStr << _timeRangeStart << "-" << _timeRangeEnd;
    return oStr.str();
  }

}
```

33.497 stdair/bom/TimePeriodKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Classes

- struct stdair::TimePeriodKey

  Key of time-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.498 stdair/bom/TimePeriodKey.hpp

```
#ifndef __STDAIR_BOM_TIMEPERIODKEY_HPP
#define __STDAIR_BOM_TIMEPERIODKEY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STDAIR
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>

namespace stdair {
  struct TimePeriodKey : public KeyAbstract {

  public:
    // /////////// Construction ///////////
    TimePeriodKey (const Time_T&,
                   const Time_T&);
    TimePeriodKey (const TimePeriodKey&);
    ~TimePeriodKey ();
  private:
    TimePeriodKey ();

  public:
    // /////////// Getter //////////
    const Time_T& getTimeRangeStart() const {
      return _timeRangeStart;
    }

    const Time_T& getTimeRangeEnd() const {
      return _timeRangeEnd;
    }

    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string toString() const;

  private:
    // ///////////////// Attributes ///////////////////
    Time_T _timeRangeStart;

    Time_T _timeRangeEnd;

  };

}
#endif // __STDAIR_BOM_TIMEPERIODKEY_HPP
```

33.499 stdair/bom/TimePeriodTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< TimePeriod * > stdair::TimePeriodList_T
- typedef std::map< const MapKey_T, TimePeriod * > stdair::TimePeriodMap_T
- typedef std::pair< MapKey_T, TimePeriod * > stdair::TimePeriodWithKey_T
- typedef std::list< TimePeriodWithKey_T > stdair::TimePeriodDetailedList_T

33.500 stdair/bom/TimePeriodTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_TIMEPERIODTYPES_HPP
#define __STDAIR_BOM_TIMEPERIODTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class TimePeriod;

  typedef std::list<TimePeriod*> TimePeriodList_T;

  typedef std::map<const MapKey_T, TimePeriod*> TimePeriodMap_T;

  typedef std::pair<MapKey_T, TimePeriod*> TimePeriodWithKey_T;
  typedef std::list<TimePeriodWithKey_T> TimePeriodDetailedList_T;
}
#endif // __STDAIR_BOM_TIMEPERIODTYPES_HPP

```

33.501 stdair/bom/TravelSolutionStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_BookingClass.hpp>
#include <stdair/bom/TravelSolutionStruct.hpp>
#include <stdair/bom/BomKeyManager.hpp>
#include <stdair/bom/ParsedKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.502 stdair/bom/TravelSolutionStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_BookingClass.hpp>
00009 #include <stdair/bom/TravelSolutionStruct.hpp>
00010 #include <stdair/bom/BomKeyManager.hpp>
00011 #include <stdair/bom/ParsedKey.hpp>
00012
00013 namespace stdair {
00014   // ////////////////////////////////////////////////////////////////////
00015   TravelSolutionStruct::TravelSolutionStruct() : _chosenFareOption (NULL) {
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   TravelSolutionStruct::~TravelSolutionStruct() {
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   void TravelSolutionStruct::toStream (std::ostream& ioOut) const {
00024     ioOut << describe();
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   void TravelSolutionStruct::fromStream (std::istream& ioIn) {
00029   }
00030
00031   // ////////////////////////////////////////////////////////////////////
00032   const std::string TravelSolutionStruct::describeSegmentPath() const {
00033     std::ostringstream oStr;
00034
00035     //
00036     oStr << "Segment path: ";
00037     unsigned short idx = 0;
00038     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00039          lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00040       if (idx != 0) {
00041         oStr << " - ";
00042       }
00043       const std::string& lSegmentPathString = *lItSegmentPath;
00044       const stdair::ParsedKey& lSegmentParsedKey =
00045         stdair::BomKeyManager::extractKeys (lSegmentPathString);
00046       const std::string& lSegmentKey = lSegmentParsedKey.toString();
00047       oStr << lSegmentKey;
00048     }
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   const std::string TravelSolutionStruct::describe() const {
00054     std::ostringstream oStr;
00055
00056     //
00057     oStr << "Segment path: ";
00058     unsigned short idx = 0;
00059     for (SegmentPath_T::const_iterator lItSegmentPath = _segmentPath.begin();
00060          lItSegmentPath != _segmentPath.end(); ++lItSegmentPath, ++idx) {
00061       if (idx != 0) {
00062         oStr << "-";
00063       }
00064       const std::string& lSegmentPathString = *lItSegmentPath;
00065       const stdair::ParsedKey& lSegmentParsedKey =
```

```
        stdair::BomKeyManager::extractKeys (lSegmentPathString);
      const std::string& lSegmentKey = lSegmentParsedKey.toString();
      oStr << lSegmentKey;
    }
    oStr << " ### ";

    //
    if (_chosenFareOption != NULL) {
      oStr << "Chosen fare option: " << _chosenFareOption->describe()
           << " ## Among: ";
    } else {
      oStr << "Fare options: ";
    }

    //
    idx = 0;
    for (FareOptionList_T::const_iterator lItFareOption= _fareOptionList.begin();
         lItFareOption != _fareOptionList.end(); ++lItFareOption, ++idx) {
      if (idx != 0) {
        oStr << " , ";
      }
      const FareOptionStruct& lFareOption = *lItFareOption;
      oStr << lFareOption.describe();
    }

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string TravelSolutionStruct::display() const {
    std::ostringstream oStr;

    // List of segment keys (one per segment)
    unsigned short idx = 0;
    for (SegmentPath_T::const_iterator itSegPath = _segmentPath.begin();
         itSegPath != _segmentPath.end(); ++itSegPath, ++idx) {
      if (idx != 0) {
        oStr << " ; ";
      }
      const std::string& lSegmentPathString = *itSegPath;
      const stdair::ParsedKey& lSegmentParsedKey =
        stdair::BomKeyManager::extractKeys (lSegmentPathString);
      const std::string& lSegmentKey = lSegmentParsedKey.toString();
      oStr << "[" << idx << "] " << lSegmentKey;
    }

    // List of fare options (for the whole O&D)
    oStr << " --- ";
    idx = 0;
    for (FareOptionList_T::const_iterator itFareOption = _fareOptionList.begin();
         itFareOption != _fareOptionList.end(); ++itFareOption, ++idx) {
      if (idx != 0) {
        oStr << " , ";
      }
      const FareOptionStruct& lFareOption = *itFareOption;
      oStr << lFareOption.display();
    }

    // List of booking class availability maps: one map per segment
    oStr << " --- ";
    idx = 0;
    for (ClassAvailabilityMapHolder_T::const_iterator itSegMap =
           _classAvailabilityMapHolder.begin();
         itSegMap != _classAvailabilityMapHolder.end(); ++itSegMap, ++idx) {
      if (idx != 0) {
```

```
        oStr << " ; ";
      }
      // Retrieve the booking class availability map
      const ClassAvailabilityMap_T& lClassAvlMap = *itSegMap;
      oStr << "[" << idx << "] ";

      // List (map) of booking class availabilities
      unsigned short jdx = 0;
      for (ClassAvailabilityMap_T::const_iterator itClass = lClassAvlMap.begin();
           itClass != lClassAvlMap.end(); ++itClass, ++jdx) {
        if (jdx != 0) {
          oStr << " ";
        }
        const ClassCode_T& lClassCode = itClass->first;
        const Availability_T& lAvl = itClass->second;
        oStr << lClassCode << ":" << lAvl;
      }
    }

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::addSegment (const std::string& iKey) {
    _segmentPath.push_back (iKey);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addClassAvailabilityMap (const ClassAvailabilityMap_T& iMap) {
    _classAvailabilityMapHolder.push_back (iMap);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addClassObjectIDMap (const ClassObjectIDMap_T& iMap) {
    _classObjectIDMapHolder.push_back (iMap);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addClassYieldMap (const ClassYieldMap_T& iMap) {
    _classYieldMapHolder.push_back (iMap);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addBidPriceVector (const BidPriceVector_T& iBpv) {
    _bidPriceVectorHolder.push_back (iBpv);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addClassBpvMap (const ClassBpvMap_T& iMap) {
    _classBpvMapHolder.push_back (iMap);
  }

  // ////////////////////////////////////////////////////////////////////
  void TravelSolutionStruct::
  addFareOption (const FareOptionStruct& iFareOption) {
    _fareOptionList.push_back (iFareOption);
  }

}
```

33.503 stdair/bom/TravelSolutionStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/FareOptionTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
```

Classes

- struct stdair::TravelSolutionStruct

  Structure holding the elements of a travel solution.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.504 stdair/bom/TravelSolutionStruct.hpp

```
#ifndef __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
#define __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
#include <stdair/bom/FareOptionStruct.hpp>
#include <stdair/bom/FareOptionTypes.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>

namespace stdair {

  struct TravelSolutionStruct : public StructAbstract {
  public:
    // /////////// Getters ///////////////
    const SegmentPath_T& getSegmentPath() const {
      return _segmentPath;
    }

    const ClassAvailabilityMapHolder_T& getClassAvailabilityMapHolder() const {
      return _classAvailabilityMapHolder;
    }

    const ClassObjectIDMapHolder_T& getClassObjectIDMapHolder() const {
      return _classObjectIDMapHolder;
    }

    const ClassYieldMapHolder_T& getClassYieldMapHolder() const {
      return _classYieldMapHolder;
    }

    const BidPriceVectorHolder_T& getBidPriceVectorHolder() const {
      return _bidPriceVectorHolder;
    }

    const ClassBpvMapHolder_T& getClassBpvMapHolder() const {
      return _classBpvMapHolder;
    }

    const FareOptionList_T& getFareOptionList() const {
      return _fareOptionList;
    }

    FareOptionList_T& getFareOptionListRef() {
      return _fareOptionList;
    }

    const FareOptionStruct& getChosenFareOption() const {
      assert (_chosenFareOption != NULL);
      return *_chosenFareOption;
    }

  public:
    // /////////// Setters ///////////////
    void addSegment (const std::string&);

    void addClassAvailabilityMap (const ClassAvailabilityMap_T&);
```

```
00080
00082     void addClassObjectIDMap (const ClassObjectIDMap_T&);
00083
00085     void addClassYieldMap (const ClassYieldMap_T&);
00086
00088     void addBidPriceVector (const BidPriceVector_T&);
00089
00091     void addClassBpvMap (const ClassBpvMap_T&);
00092
00094     void addFareOption (const FareOptionStruct&);
00095
00097     void setChosenFareOption (const FareOptionStruct& iChosenFO) {
00098       _chosenFareOption = &iChosenFO;
00099     }
00100
00101
00102   public:
00103     // /////////// Display support method /////////////
00109     void toStream (std::ostream& ioOut) const;
00110
00115     void fromStream (std::istream& ioIn);
00116
00120     const std::string describe() const;
00121
00125     const std::string display() const;
00126
00130     const std::string describeSegmentPath() const;
00131
00132
00133   public:
00134     // ///////////// Constructors & Destructor ///////////////
00138     TravelSolutionStruct();
00139
00143     ~TravelSolutionStruct();
00144
00145
00146   private:
00147     // /////////////////////// Attributes ///////////////////////
00151     SegmentPath_T _segmentPath;
00152
00156     ClassAvailabilityMapHolder_T _classAvailabilityMapHolder;
00157
00161     ClassObjectIDMapHolder_T _classObjectIDMapHolder;
00162
00166     ClassYieldMapHolder_T _classYieldMapHolder;
00167
00171     BidPriceVectorHolder_T _bidPriceVectorHolder;
00172
00176     ClassBpvMapHolder_T _classBpvMapHolder;
00177
00181     FareOptionList_T _fareOptionList;
00182
00186     const FareOptionStruct* _chosenFareOption;
00187   };
00188
00189 }
00190 #endif // __STDAIR_BOM_TRAVELSOLUTIONSTRUCT_HPP
```

33.505 stdair/bom/TravelSolutionTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomIDTypes.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< TravelSolutionStruct > stdair::TravelSolutionList_T
- typedef KeyList_T stdair::SegmentPath_T
- typedef std::list< SegmentPath_T > stdair::SegmentPathList_T
- typedef std::map< const ClassCode_T, Availability_T > stdair::ClassAvailabilityMap_T
- typedef std::list< ClassAvailabilityMap_T > stdair::ClassAvailabilityMapHolder_T
- typedef std::map< const ClassCode_T, BookingClassID_T > stdair::ClassObjectIDMap_T
- typedef std::list< ClassObjectIDMap_T > stdair::ClassObjectIDMapHolder_T
- typedef std::map< const ClassCode_T, YieldValue_T > stdair::ClassYieldMap_T
- typedef std::list< ClassYieldMap_T > stdair::ClassYieldMapHolder_T
- typedef std::list< BidPriceVector_T > stdair::BidPriceVectorHolder_T
- typedef std::map< const ClassCode_T, const BidPriceVector_T * > stdair::ClassBpvMap_T
- typedef std::list< ClassBpvMap_T > stdair::ClassBpvMapHolder_T

33.506 stdair/bom/TravelSolutionTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
#define __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <list>
#include <map>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/bom/key_types.hpp>
#include <stdair/stdair_inventory_types.hpp> // bid price related types.
#include <stdair/bom/BomIDTypes.hpp>

namespace stdair {

  // Forward declarations.
  struct TravelSolutionStruct;

  typedef std::list<TravelSolutionStruct> TravelSolutionList_T;

  typedef KeyList_T SegmentPath_T;

  typedef std::list<SegmentPath_T> SegmentPathList_T;

  typedef std::map<const ClassCode_T, Availability_T> ClassAvailabilityMap_T;

  typedef std::list<ClassAvailabilityMap_T> ClassAvailabilityMapHolder_T;

  typedef std::map<const ClassCode_T, BookingClassID_T> ClassObjectIDMap_T;

  typedef std::list<ClassObjectIDMap_T> ClassObjectIDMapHolder_T;

  typedef std::map<const ClassCode_T, YieldValue_T> ClassYieldMap_T;

  typedef std::list<ClassYieldMap_T> ClassYieldMapHolder_T;

  typedef std::list<BidPriceVector_T> BidPriceVectorHolder_T;

  typedef std::map<const ClassCode_T, const BidPriceVector_T*> ClassBpvMap_T;

  typedef std::list<ClassBpvMap_T> ClassBpvMapHolder_T;
}
#endif // __STDAIR_BOM_TRAVELSOLUTIONTYPES_HPP
```

33.507 stdair/bom/VirtualClassStruct.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/VirtualClassStruct.hpp>
#include <stdair/bom/BookingClass.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.508 stdair/bom/VirtualClassStruct.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/bom/VirtualClassStruct.hpp>
00009 #include <stdair/bom/BookingClass.hpp>
00010
00011 namespace stdair {
00012
00013   // ////////////////////////////////////////////////////////////////////
00014   VirtualClassStruct::VirtualClassStruct() {
00015     assert (false);
00016   }
00017
00018   // ////////////////////////////////////////////////////////////////////
00019   VirtualClassStruct::VirtualClassStruct (const VirtualClassStruct& iVC)
00020     : _bookingClassList (iVC._bookingClassList), _yield (iVC._yield),
00021       _mean (iVC._mean), _stdDev (iVC._stdDev) {
00022   }
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   VirtualClassStruct::
00026   VirtualClassStruct (const BookingClassList_T& ioBookingClassList) {
00027     _bookingClassList = ioBookingClassList;
00028   }
00029
00030   // ////////////////////////////////////////////////////////////////////
00031   VirtualClassStruct::~VirtualClassStruct() {
00032
00033   }
00034
00035   // ////////////////////////////////////////////////////////////////////
00036   void VirtualClassStruct::toStream (std::ostream& ioOut) const {
00037     ioOut << describe();
00038   }
00039
00040   // ////////////////////////////////////////////////////////////////////
00041   void VirtualClassStruct::fromStream (std::istream& ioIn) {
00042   }
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   const std::string VirtualClassStruct::describe() const {
00046     std::ostringstream oStr;
00047     oStr << "Yield: " << _yield
00048          << ", Demand N (" << _mean << ", " << _stdDev << ")";
00049     return oStr.str();
00050   }
00051
00052   // ////////////////////////////////////////////////////////////////////
00053   const GeneratedDemandVector_T VirtualClassStruct::
00054   getGeneratedDemandVector() const  {
00055       GeneratedDemandVector_T lDemandVector;
00056       const bool isBookingClassListEmpty = _bookingClassList.empty();
00057       if (isBookingClassListEmpty == false) {
00058         assert (isBookingClassListEmpty == false);
00059         BookingClassList_T::const_iterator itBC = _bookingClassList.begin();
00060         BookingClass* lBC_ptr = *itBC;
00061         const GeneratedDemandVector_T& lFirstDemandVector =
00062           lBC_ptr->getGeneratedDemandVector();
00063         const unsigned int lFirstDemandVectorSize = lFirstDemandVector.size();
00064         for (unsigned int i = 0; i < lFirstDemandVectorSize; ++i) {
00065           const double& lDemand = lFirstDemandVector[i];
```

```
          lDemandVector.push_back(lDemand);
        }
        const unsigned int& lDemandVectorSize = lDemandVector.size();
        ++itBC;
        for (; itBC != _bookingClassList.end(); ++ itBC) {
          lBC_ptr = *itBC;
          assert(lBC_ptr != NULL);
          const GeneratedDemandVector_T& lCurrentDemandVector =
            lBC_ptr->getGeneratedDemandVector();
          const unsigned int& lCurrentDemandVectorSize =
            lCurrentDemandVector.size();
          assert(lDemandVectorSize == lCurrentDemandVectorSize);
          for (unsigned int i = 0; i < lDemandVectorSize; ++i) {
            lDemandVector[i] += lCurrentDemandVector[i];
          }
        }
      }
      return lDemandVector;
  }
}
```

33.509 stdair/bom/VirtualClassStruct.hpp File Reference

```
#include <iosfwd>
#include <string>
#include <vector>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>
```

Classes

- struct stdair::VirtualClassStruct

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.510 stdair/bom/VirtualClassStruct.hpp

```
#ifndef __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
#define __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <string>
#include <vector>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/basic/StructAbstract.hpp>
#include <stdair/bom/BookingClassTypes.hpp>

namespace stdair {
  // Forward declarations.
  class BookingClass;

  struct VirtualClassStruct : public StructAbstract {
  public:
    // /////////// Getters ///////////////
    const BookingClassList_T& getBookingClassList() const {
      return _bookingClassList;
    }

    const Yield_T& getYield() const {
      return _yield;
    }

    const MeanValue_T& getMean() const {
      return _mean;
    }

    const StdDevValue_T& getStdDev() const {
      return _stdDev;
    }

    const BookingLimit_T& getCumulatedBookingLimit () const {
      return _cumulatedBookingLimit;
    }

    const ProtectionLevel_T& getCumulatedProtection () const {
      return _cumulatedProtection;
    }

    const GeneratedDemandVector_T getGeneratedDemandVector () const;

  public:
    // /////////// Setters ///////////////
    void setYield (const Yield_T& iYield) {
      _yield = iYield;
    }

    void setMean (const MeanValue_T& iMean) {
      _mean = iMean;
    }

    void setStdDev (const StdDevValue_T& iStdDev) {
      _stdDev = iStdDev;
    }

```

```
    void setCumulatedBookingLimit (const BookingLimit_T& iBL) {
      _cumulatedBookingLimit = iBL;
    }

    void setCumulatedProtection (const ProtectionLevel_T& iP) {
      _cumulatedProtection = iP;
    }

    void addBookingClass (BookingClass& iBookingClass) {
      _bookingClassList.push_back(&iBookingClass);
    }

  public:
    // /////////// Display support method /////////////
    void toStream (std::ostream& ioOut) const;

    void fromStream (std::istream& ioIn);

    const std::string describe() const;


  public:
    // ///////////// Constructors & Destructor ///////////////
    VirtualClassStruct (const VirtualClassStruct&);
    VirtualClassStruct (const BookingClassList_T&);
    ~VirtualClassStruct();

  private:
    VirtualClassStruct();


  private:
    // ////////////////////// Attributes //////////////////////
    BookingClassList_T _bookingClassList;

    Yield_T _yield;

    MeanValue_T _mean;

    StdDevValue_T _stdDev;

    BookingLimit_T _cumulatedBookingLimit;

    ProtectionLevel_T _cumulatedProtection;
  };

}
#endif // __STDAIR_BOM_VIRTUALCLASSSTRUCT_HPP
```

33.511 stdair/bom/VirtualClassTypes.hpp File Reference

```
#include <list>
#include <map>
#include <stdair/stdair_basic_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< VirtualClassStruct > stdair::VirtualClassList_T
- typedef std::map< const YieldLevel_T, VirtualClassStruct > stdair::VirtualClassMap_T

33.512 stdair/bom/VirtualClassTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_VIRTUALCLASSTYPES_HPP
#define __STDAIR_BOM_VIRTUALCLASSTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <list>
#include <map>
// STDAIR
#include <stdair/stdair_basic_types.hpp>

namespace stdair {

  // Forward declarations.
  struct VirtualClassStruct;

  typedef std::list<VirtualClassStruct> VirtualClassList_T;

  typedef std::map<const YieldLevel_T, VirtualClassStruct> VirtualClassMap_T;
}
#endif // __STDAIR_BOM_VIRTUALCLASSTYPES_HPP

```

33.513 stdair/bom/YieldFeatures.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/bom/YieldFeatures.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.514 stdair/bom/YieldFeatures.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // StdAir
00008 #include <stdair/basic/BasConst_Request.hpp>
00009 #include <stdair/service/Logger.hpp>
00010 #include <stdair/bom/YieldFeatures.hpp>
00011
00012 namespace stdair {
00013
00014   // ////////////////////////////////////////////////////////////////////
00015   YieldFeatures::YieldFeatures()
00016     : _key (TRIP_TYPE_ONE_WAY,
00017            DEFAULT_PREFERRED_CABIN),
00018       _parent (NULL)  {
00019     // That constructor is used by the serialisation process
00020   }
00021
00022   // ////////////////////////////////////////////////////////////////////
00023   YieldFeatures::YieldFeatures (const YieldFeatures& iFeatures)
00024     : _key (iFeatures.getKey()), _parent (NULL)  {
00025   }
00026
00027   // ////////////////////////////////////////////////////////////////////
00028   YieldFeatures::YieldFeatures (const Key_T& iKey)
00029     : _key (iKey), _parent (NULL)  {
00030   }
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   YieldFeatures::~YieldFeatures () {
00034   }
00035
00036   // ////////////////////////////////////////////////////////////////////
00037   std::string YieldFeatures::toString() const {
00038     std::ostringstream oStr;
00039     oStr << describeKey();
00040     return oStr.str();
00041   }
00042
00043   // ////////////////////////////////////////////////////////////////////
00044   bool YieldFeatures::
00045   isTripTypeValid (const TripType_T& iBookingRequestTripType) const {
00046     bool oIsTripTypeValidFlag = true;
00047
00048     // Check whether the yield trip type is the same as the booking request
00049     // trip type
00050     const TripType_T& lYieldTripType = getTripType();
00051     if (iBookingRequestTripType == lYieldTripType) {
00052       // One way case
00053       return oIsTripTypeValidFlag;
00054     }
00055
00056     if (iBookingRequestTripType == TRIP_TYPE_INBOUND ||
00057         iBookingRequestTripType == TRIP_TYPE_OUTBOUND) {
00058       // Round trip case.
00059       if (lYieldTripType == TRIP_TYPE_ROUND_TRIP) {
00060         return oIsTripTypeValidFlag;
00061       }
00062     }
00063
00064     oIsTripTypeValidFlag = false;
00065     return false;
```

```
00066   }
00067
00068 }
00069
```

33.515 stdair/bom/YieldFeatures.hpp File Reference

```
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
#include <stdair/bom/YieldFeaturesTypes.hpp>
```

Classes

- class stdair::YieldFeatures

  Class representing the actual attributes for a yield date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.516 stdair/bom/YieldFeatures.hpp

```
00001 #ifndef __STDAIR_BOM_YIELDFEATURES_HPP
00002 #define __STDAIR_BOM_YIELDFEATURES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/BomAbstract.hpp>
00009 #include <stdair/bom/YieldFeaturesKey.hpp>
00010 #include <stdair/bom/YieldFeaturesTypes.hpp>
00011
00012 // Forward declaration
00013 namespace stdair {
00014
00019   class YieldFeatures : public BomAbstract {
00020     template <typename BOM> friend class FacBom;
00021     template <typename BOM> friend class FacCloneBom;
00022     friend class FacBomManager;
00023
00024   public:
00025     // /////// Type definitions
00029     typedef YieldFeaturesKey Key_T;
00030
00031   public:
00032     // /////////// Display support methods /////////
00038     void toStream (std::ostream& ioOut) const {
00039       ioOut << toString();
00040     }
00041
00047     void fromStream (std::istream& ioIn) {
00048     }
00049
00053     std::string toString() const;
00054
00058     const std::string describeKey() const {
00059       return _key.toString();
00060     }
00061
00062   public:
00063     // ////////// Getters ////////////
00067     const Key_T& getKey() const {
00068       return _key;
00069     }
00070
00074     BomAbstract* const getParent() const {
00075       return _parent;
00076     }
00077
00081     const HolderMap_T& getHolderMap() const {
00082       return _holderMap;
00083     }
00084
00088     const CabinCode_T& getCabinCode() const {
00089       return _key.getCabinCode();
00090     }
00091
00095     const TripType_T& getTripType() const {
00096       return _key.getTripType();
00097     }
00098
00099
00100   public:
00101     // ////////////// Business methods ///////////////
00106     bool isTripTypeValid (const TripType_T&) const;
00107
```

```
00108
00109   protected:
00110     // ////////// Constructors and destructors /////////
00114     YieldFeatures (const Key_T&);
00115
00119     virtual ~YieldFeatures();
00120
00121   private:
00125     YieldFeatures();
00126
00130     YieldFeatures (const YieldFeatures&);
00131
00132
00133   protected:
00134     // ////////// Attributes /////////
00138     Key_T _key;
00139
00143     BomAbstract* _parent;
00144
00148     HolderMap_T _holderMap;
00149   };
00150
00151 }
00152 #endif // __STDAIR_BOM_YIELDFEATURES_HPP
00153
```

33.517 stdair/bom/YieldFeaturesKey.cpp File Reference

```
#include <ostream>
#include <sstream>
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.518 stdair/bom/YieldFeaturesKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <ostream>
#include <sstream>
// StdAir
#include <stdair/basic/BasConst_Request.hpp>
#include <stdair/bom/YieldFeaturesKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  YieldFeaturesKey::YieldFeaturesKey()
    : _tripType (TRIP_TYPE_ONE_WAY),
      _cabinCode (DEFAULT_PREFERRED_CABIN) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  YieldFeaturesKey::YieldFeaturesKey (const stdair::TripType_T& iTripType,
                                      const stdair::CabinCode_T& iCabin)
    : _tripType (iTripType), _cabinCode (iCabin) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldFeaturesKey::YieldFeaturesKey (const YieldFeaturesKey& iKey)
    : _tripType (iKey.getTripType()), _cabinCode (iKey.getCabinCode()) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldFeaturesKey::~YieldFeaturesKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldFeaturesKey::toStream (std::ostream& ioOut) const {
    ioOut << "YieldFeaturesKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldFeaturesKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string YieldFeaturesKey::toString() const {
    std::ostringstream oStr;
    oStr << _tripType << " -- " << _cabinCode;
    return oStr.str();
  }

}
```

33.519 stdair/bom/YieldFeaturesKey.hpp File Reference

```
#include <stdair/bom/KeyAbstract.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Classes

- struct stdair::YieldFeaturesKey

  Key of date-period.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.520 stdair/bom/YieldFeaturesKey.hpp

```
00001 #ifndef __STDAIR_BOM_YIELDFEATURESKEY_HPP
00002 #define __STDAIR_BOM_YIELDFEATURESKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/bom/KeyAbstract.hpp>
00009 #include <stdair/stdair_date_time_types.hpp>
00010 #include <stdair/stdair_demand_types.hpp>
00011 #include <stdair/stdair_inventory_types.hpp>
00012
00013 namespace stdair  {
00014
00018   struct YieldFeaturesKey : public KeyAbstract {
00019   public:
00020     // /////////// Construction ///////////
00024     YieldFeaturesKey (const TripType_T&, const CabinCode_T&);
00028     YieldFeaturesKey (const YieldFeaturesKey&);
00032     ~YieldFeaturesKey ();
00033   private:
00037     YieldFeaturesKey ();
00038
00039   public:
00040     // /////////// Getters //////////
00044     const TripType_T& getTripType() const {
00045       return _tripType;
00046     }
00047
00051     const CabinCode_T& getCabinCode() const {
00052       return _cabinCode;
00053     }
00054
00055   public:
00056     // /////////// Display support methods /////////
00061     void toStream (std::ostream& ioOut) const;
00062
00067     void fromStream (std::istream& ioIn);
00068
00074     const std::string toString() const;
00075
00076   private:
00077     // /////////////// Attributes /////////////////
00081     TripType_T _tripType;
00082
00086     CabinCode_T _cabinCode;
00087   };
00088
00089 }
00090 #endif // __STDAIR_BOM_YIELDFEATURESKEY_HPP
```

33.521 stdair/bom/YieldFeaturesTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< YieldFeatures * > stdair::YieldFeaturesList_T
- typedef std::map< const MapKey_T, YieldFeatures * > stdair::YieldFeaturesMap_T
- typedef std::pair< MapKey_T, YieldFeatures * > stdair::YieldFeaturesWithKey_T
- typedef std::list< YieldFeaturesWithKey_T > stdair::YieldFeaturesDetailedList_T

33.522 stdair/bom/YieldFeaturesTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_YIELDFEATURESTYPES_HPP
#define __STDAIR_BOM_YIELDFEATURESTYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// STDAIR
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class YieldFeatures;

  typedef std::list<YieldFeatures*> YieldFeaturesList_T;

  typedef std::map<const MapKey_T, YieldFeatures*> YieldFeaturesMap_T;

  typedef std::pair<MapKey_T, YieldFeatures*> YieldFeaturesWithKey_T;
  typedef std::list<YieldFeaturesWithKey_T> YieldFeaturesDetailedList_T;
}
#endif // __STDAIR_BOM_YIELDFEATURESTYPES_HPP

```

33.523 stdair/bom/YieldStore.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/bom/YieldStore.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.524 stdair/bom/YieldStore.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// StdAir
#include <stdair/bom/YieldStore.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  YieldStore::YieldStore (const Key_T& iKey) : _key (iKey), _parent (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldStore::~YieldStore () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string YieldStore::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

}

```

33.525 stdair/bom/YieldStore.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldStoreKey.hpp>
#include <stdair/bom/YieldStoreTypes.hpp>
```

Classes

- class stdair::YieldStore

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.526 stdair/bom/YieldStore.hpp

```
#ifndef __STDAIR_BOM_YIELDSTORE_HPP
#define __STDAIR_BOM_YIELDSTORE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/bom/YieldStoreKey.hpp>
#include <stdair/bom/YieldStoreTypes.hpp>

namespace stdair {

  class YieldStore : public BomAbstract {
    template <typename BOM> friend class FacBom;
    friend class FacBomManager;

  public :
    // Type definitions
    typedef YieldStoreKey Key_T;

  public:
    // /////////// Display support methods /////////
    void toStream (std::ostream& ioOut) const { ioOut << toString(); }

    BomAbstract* const getParent() const { return _parent; }

    void fromStream (std::istream& ioIn) { }

    std::string toString() const;

    const std::string describeKey() const { return _key.toString(); }

  public:
    // ////////// Getters ////////////
    const Key_T& getKey() const { return _key; }

    const AirlineCode_T& getAirlineCode () const {
      return _key.getAirlineCode();
    }

  protected:
    YieldStore (const Key_T&);
    YieldStore (const YieldStore&);
    ~YieldStore();

  protected:
    // Attributes
    Key_T _key;
    BomAbstract* _parent;
  };

}
#endif // __STDAIR_BOM_YIELDSTORE_HPP
```

33.527 stdair/bom/YieldStoreKey.cpp File Reference

```
#include <stdair/bom/YieldStoreKey.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.528 stdair/bom/YieldStoreKey.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/bom/YieldStoreKey.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  YieldStoreKey::YieldStoreKey (const AirlineCode_T& iAirlineCode)
    : _airlineCode (iAirlineCode) {
  }
  // ////////////////////////////////////////////////////////////////////
  YieldStoreKey::YieldStoreKey (const YieldStoreKey& iKey)
    : _airlineCode (iKey._airlineCode) {
  }

  // ////////////////////////////////////////////////////////////////////
  YieldStoreKey::~YieldStoreKey () {
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldStoreKey::toStream (std::ostream& ioOut) const {
    ioOut << "YieldStoreKey: " << toString() << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void YieldStoreKey::fromStream (std::istream& ioIn) {
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string YieldStoreKey::toString() const {
    std::ostringstream oStr;
    oStr << _airlineCode;
    return oStr.str();
  }

}
```

33.529 stdair/bom/YieldStoreKey.hpp File Reference

```
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/KeyAbstract.hpp>
```

Classes

- struct stdair::YieldStoreKey

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.530 stdair/bom/YieldStoreKey.hpp

```
00001 #ifndef __STDAIR_BOM_YIELDSTOREKEY_HPP
00002 #define __STDAIR_BOM_YIELDSTOREKEY_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // StdAir
00008 #include <stdair/stdair_inventory_types.hpp>
00009 #include <stdair/bom/KeyAbstract.hpp>
00010
00011 namespace stdair {
00012
00014   struct YieldStoreKey : public KeyAbstract {
00015
00016   private:
00017     // /////////// Default constructor //////////
00018     YieldStoreKey () { };
00019
00020   public:
00021     // /////////// Construction ///////////
00023     YieldStoreKey (const AirlineCode_T& iAirlineCode);
00024     YieldStoreKey (const YieldStoreKey&);
00026     ~YieldStoreKey ();
00027
00028     // /////////// Getters //////////
00030     const AirlineCode_T& getAirlineCode() const {
00031       return _airlineCode;
00032     }
00033
00034     // /////////// Display support methods /////////
00037     void toStream (std::ostream& ioOut) const;
00038
00041     void fromStream (std::istream& ioIn);
00042
00048     const std::string toString() const;
00049
00050   private:
00051     // Attributes
00053     AirlineCode_T _airlineCode;
00054   };
00055
00056 }
00057 #endif // __STDAIR_BOM_YIELDSTOREKEY_HPP
```

33.531 stdair/bom/YieldStoreTypes.hpp File Reference

```
#include <map>
#include <list>
#include <stdair/bom/key_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::list< YieldStore * > stdair::YieldStoreList_T
- typedef std::map< const MapKey_T, YieldStore * > stdair::YieldStoreMap_T

33.532 stdair/bom/YieldStoreTypes.hpp

```
// ////////////////////////////////////////////////////////////////////
#ifndef __STDAIR_BOM_YIELDSTORETYPES_HPP
#define __STDAIR_BOM_YIELDSTORETYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <map>
#include <list>
// StdAir
#include <stdair/bom/key_types.hpp>

namespace stdair {

  // Forward declarations.
  class YieldStore;

  typedef std::list<YieldStore*> YieldStoreList_T;

  typedef std::map<const MapKey_T, YieldStore*> YieldStoreMap_T;

}
#endif // __STDAIR_BOM_YIELDSTORETYPES_HPP

```

33.533 stdair/command/CmdAbstract.cpp File Reference

```
#include <stdair/command/CmdAbstract.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.534 stdair/command/CmdAbstract.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/command/CmdAbstract.hpp>

namespace stdair {

}
```

33.535 stdair/command/CmdAbstract.hpp File Reference

Classes

- class stdair::CmdAbstract

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.536 stdair/command/CmdAbstract.hpp

```
#ifndef __STDAIR_CMD_CMDABSTRACT_HPP
#define __STDAIR_CMD_CMDABSTRACT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////

namespace stdair {

  class CmdAbstract {
  public:

  };

}
#endif // __STDAIR_CMD_CMDABSTRACT_HPP
```

33.537 stdair/command/CmdBomManager.cpp File Reference

33.538 stdair/command/CmdBomManager.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/basic/BasConst_General.hpp>
00013 #include <stdair/basic/BasConst_DefaultObject.hpp>
00014 #include <stdair/basic/BasConst_Request.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017 #include <stdair/bom/BomRoot.hpp>
00018 #include <stdair/bom/Inventory.hpp>
00019 #include <stdair/bom/AirlineFeature.hpp>
00020 #include <stdair/bom/FlightDate.hpp>
00021 #include <stdair/bom/LegDate.hpp>
00022 #include <stdair/bom/LegCabin.hpp>
00023 #include <stdair/bom/SegmentDate.hpp>
00024 #include <stdair/bom/SegmentCabin.hpp>
00025 #include <stdair/bom/FareFamily.hpp>
00026 #include <stdair/bom/BookingClass.hpp>
00027 #include <stdair/bom/AirportPair.hpp>
00028 #include <stdair/bom/PosChannel.hpp>
00029 #include <stdair/bom/DatePeriod.hpp>
00030 #include <stdair/bom/TimePeriod.hpp>
00031 #include <stdair/bom/FareFeatures.hpp>
00032 #include <stdair/bom/YieldFeatures.hpp>
00033 #include <stdair/bom/AirlineClassList.hpp>
00034 #include <stdair/bom/BomManager.hpp>
00035 #include <stdair/bom/TravelSolutionStruct.hpp>
00036 #include <stdair/bom/BookingRequestStruct.hpp>
00037 #include <stdair/factory/FacBomManager.hpp>
00038 #include <stdair/factory/FacBom.hpp>
00039 #include <stdair/command/CmdBomManager.hpp>
00040 #include <stdair/service/Logger.hpp>
00041 #include <stdair/bom/OnDDate.hpp>
00042 #include <stdair/bom/SegmentPeriod.hpp>
00043 #include <stdair/bom/FlightPeriod.hpp>
00044
00045 namespace stdair {
00046
00047   // ////////////////////////////////////////////////////////////////////
00048   void CmdBomManager::buildSampleBom (BomRoot& ioBomRoot) {
00049
00050     // DEBUG
00051     STDAIR_LOG_DEBUG ("StdAir is building the BOM tree from built-in "
00052                       << "specifications.");
00053
00054     // ////// Basic Bom Tree ///////
00055     // Build the inventory (flight-dates) and the schedule (flight period)
00056     // parts.
00057     buildSampleInventorySchedule (ioBomRoot);
00058
00059     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00060     buildSamplePricing (ioBomRoot);
00061
00062     // ////// Partnership Bom Tree ///////
00063     // Build the inventory (flight-dates) and the schedule (flight period)
00064     // parts.
00065     buildPartnershipsSampleInventoryAndRM (ioBomRoot);
00066
00067     // Build the pricing (fare rules) and revenue accounting (yields) parts.
00068     buildPartnershipsSamplePricing (ioBomRoot);
```

```

  // Build a dummy inventory, needed by RMOL.
  buildCompleteDummyInventory (ioBomRoot);

  // ////// Fare Families Bom Tree ///////
  // Build the inventory (flight-dates) and the schedule (flight period)
  // parts with fare families.
  buildSampleInventoryScheduleForFareFamilies (ioBomRoot);

  // Build the pricing (fare rules) and revenue accounting (yields) parts.
  buildSamplePricingForFareFamilies (ioBomRoot);

  // Build a dummy inventory, needed by RMOL.
  buildCompleteDummyInventoryForFareFamilies (ioBomRoot);
}

// ////////////////////////////////////////////////////////////////////
void CmdBomManager::buildSampleInventorySchedule (BomRoot& ioBomRoot) {

  // Inventory
  // Step 0.1: Inventory level
  // Create an Inventory for BA
  const AirlineCode_T lAirlineCodeBA ("BA");
  const InventoryKey lBAKey (lAirlineCodeBA);
  Inventory& lBAInv = FacBom<Inventory>::instance().create (lBAKey);
  FacBomManager::addToListAndMap (ioBomRoot, lBAInv);
  FacBomManager::linkWithParent (ioBomRoot, lBAInv);

  // Add the airline feature object to the BA inventory
  const AirlineFeatureKey lAirlineFeatureBAKey (lAirlineCodeBA);
  AirlineFeature& lAirlineFeatureBA =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureBAKey);
  FacBomManager::setAirlineFeature (lBAInv, lAirlineFeatureBA);
  FacBomManager::linkWithParent (lBAInv, lAirlineFeatureBA);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureBA);

  // Create an Inventory for AF
  const AirlineCode_T lAirlineCodeAF ("AF");
  const InventoryKey lAFKey (lAirlineCodeAF);
  Inventory& lAFInv = FacBom<Inventory>::instance().create (lAFKey);
  FacBomManager::addToListAndMap (ioBomRoot, lAFInv);
  FacBomManager::linkWithParent (ioBomRoot, lAFInv);

  // Add the airline feature object to the AF inventory
  const AirlineFeatureKey lAirlineFeatureAFKey (lAirlineCodeAF);
  AirlineFeature& lAirlineFeatureAF =
    FacBom<AirlineFeature>::instance().create (lAirlineFeatureAFKey);
  FacBomManager::setAirlineFeature (lAFInv, lAirlineFeatureAF);
  FacBomManager::linkWithParent (lAFInv, lAirlineFeatureAF);
  // Link the airline feature object with the top of the BOM tree
  FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureAF);

  // BA
  // Step 0.2: Flight-date level
  // Create a FlightDate (BA9/10-JUN-2011) for BA's Inventory
  FlightNumber_T lFlightNumber = 9;
  Date_T lDate (2011, 6, 10);
  FlightDateKey lFlightDateKey (lFlightNumber, lDate);

  FlightDate& lBA9_20110610_FD =
    FacBom<FlightDate>::instance().create (lFlightDateKey);
  FacBomManager::addToListAndMap (lBAInv, lBA9_20110610_FD);
  FacBomManager::linkWithParent (lBAInv, lBA9_20110610_FD);

  // Display the flight-date
  // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());
```

```

    // Step 0.3: Segment-date level
    // Create a first SegmentDate (LHR-SYD) for BA's Inventory
    // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=0009&from=LHR&to=SYD&depDate=100611&SellingClass=O
    const AirportCode_T lLHR ("LHR");
    const AirportCode_T lSYD ("SYD");
    const DateOffset_T l1Day (1);
    const DateOffset_T l2Days (2);
    const Duration_T l2135 (21, 45, 0);
    const Duration_T l0610 (6, 10, 0);
    const Duration_T l2205 (22, 05, 0);
    SegmentDateKey lSegmentDateKey (lLHR, lSYD);

    SegmentDate& lLHRSYDSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRSYDSegment);
    FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRSYDSegment);

    // Add the routing leg keys to the LHR-SYD segment.
    const std::string lBALHRRoutingLegStr = "BA;9;2011-Jun-10;LHR";
    const std::string lBABKKRoutingLegStr = "BA;9;2011-Jun-10;BKK";
    lLHRSYDSegment.addLegKey (lBALHRRoutingLegStr);
    lLHRSYDSegment.addLegKey (lBABKKRoutingLegStr);

    // Fill the SegmentDate content
    lLHRSYDSegment.setBoardingDate (lDate);
    lLHRSYDSegment.setOffDate (lDate + l2Days);
    lLHRSYDSegment.setBoardingTime (l2135);
    lLHRSYDSegment.setOffTime (l0610);
    lLHRSYDSegment.setElapsedTime (l2135);

    // Display the segment-date
    // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRSYDSegment);

    // Create a second SegmentDate (LHR-BKK) for BA's Inventory
    // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=0009&from=LHR&to=BKK&depDate=100611&SellingClass=O
    const AirportCode_T lBKK ("BKK");
    const Duration_T l1540 (15, 40, 0);
    const Duration_T l1105 (11, 5, 0);
    lSegmentDateKey = SegmentDateKey (lLHR, lBKK);

    SegmentDate& lLHRBKKSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRBKKSegment);
    FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRBKKSegment);

    // Add the routing leg key to the LHR-BKK segment.
    lLHRBKKSegment.addLegKey (lBALHRRoutingLegStr);

    // Fill the SegmentDate content
    lLHRBKKSegment.setBoardingDate (lDate);
    lLHRBKKSegment.setOffDate (lDate + l1Day);
    lLHRBKKSegment.setBoardingTime (l2135);
    lLHRBKKSegment.setOffTime (l1540);
    lLHRBKKSegment.setElapsedTime (l1105);

    // Display the segment-date
    // STDAIR_LOG_DEBUG ("SegmentDate: " << lLHRBKKSegment);

    // Create a third SegmentDate (BKK-SYD) for BA's Inventory
    // See http://www.britishairways.com/travel/flightinformation/public/fr_fr?&Carrier=BA&FlightNumber=0009&from=BKK&to=SYD&depDate=110611&SellingClass=O
    const Duration_T l1705 (17, 5, 0);
    const Duration_T l0905 (9, 5, 0);
    lSegmentDateKey = SegmentDateKey (lBKK, lSYD);
```

```

    SegmentDate& lBKKSYDSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKSYDSegment);
    FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKSYDSegment);

    // Add the routing leg key to the BKK-SYD segment.
    lBKKSYDSegment.addLegKey (lBABKKRoutingLegStr);

    // Fill the SegmentDate content
    lBKKSYDSegment.setBoardingDate (lDate + l1Day);
    lBKKSYDSegment.setOffDate (lDate + l2Days);
    lBKKSYDSegment.setBoardingTime (l1705);
    lBKKSYDSegment.setOffTime (l1540);
    lBKKSYDSegment.setElapsedTime (l0905);

    // Display the segment-date
    // STDAIR_LOG_DEBUG ("SegmentDate: " << lBKKSYDSegment);

    // Step 0.4: Leg-date level
    // Create a first LegDate (LHR) for BA's Inventory
    LegDateKey lLegDateKey (lLHR);

    LegDate& lLHRLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lBA9_20110610_FD, lLHRLeg);
    FacBomManager::linkWithParent (lBA9_20110610_FD, lLHRLeg);

    // Fill the LegDate content
    lLHRLeg.setOffPoint (lBKK);
    lLHRLeg.setBoardingDate (lDate);
    lLHRLeg.setOffDate (lDate + l1Day);
    lLHRLeg.setBoardingTime (l2135);
    lLHRLeg.setOffTime (l1540);
    lLHRLeg.setElapsedTime (l1105);

    // Display the leg-date
    // STDAIR_LOG_DEBUG ("LegDate: " << lLHRLeg.toString());

    // Create a second LegDate (BKK)
    lLegDateKey = LegDateKey (lBKK);

    LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lBA9_20110610_FD, lBKKLeg);
    FacBomManager::linkWithParent (lBA9_20110610_FD, lBKKLeg);

    // Display the leg-date
    // STDAIR_LOG_DEBUG ("LegDate: " << lBKKLeg.toString());

    // Fill the LegDate content
    lBKKLeg.setOffPoint (lSYD);
    lBKKLeg.setBoardingDate (lDate + l1Day);
    lBKKLeg.setOffDate (lDate + l2Days);
    lBKKLeg.setBoardingTime (l1705);
    lBKKLeg.setOffTime (l1540);
    lBKKLeg.setElapsedTime (l0905);

    // Step 0.5: segment-cabin level
    // Create a SegmentCabin (Y) for the Segment LHR-BKK of BA's Inventory
    const CabinCode_T lY ("Y");
    SegmentCabinKey lYSegmentCabinKey (lY);

    SegmentCabin& lLHRBKKSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lLHRBKKSegment, lLHRBKKSegmentYCabin);
    FacBomManager::linkWithParent (lLHRBKKSegment, lLHRBKKSegmentYCabin);

    // Display the segment-cabin
```

```
    // STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRBKKSegmentYCabin.toString());

    // Create a SegmentCabin (Y) of the Segment BKK-SYD;
    SegmentCabin& lBKKSYDSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lBKKSYDSegment, lBKKSYDSegmentYCabin);
    FacBomManager::linkWithParent (lBKKSYDSegment, lBKKSYDSegmentYCabin);


    // Display the segment-cabin
    // STDAIR_LOG_DEBUG ("SegmentCabin: " << lBKKSYDSegmentYCabin.toString());

    // Create a SegmentCabin (Y) of the Segment LHR-SYD;
    SegmentCabin& lLHRSYDSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lLHRSYDSegment, lLHRSYDSegmentYCabin);
    FacBomManager::linkWithParent (lLHRSYDSegment, lLHRSYDSegmentYCabin);

    // Display the segment-cabin
    // STDAIR_LOG_DEBUG ("SegmentCabin: " << lLHRSYDSegmentYCabin.toString());

    // Step 0.6: leg-cabin level
    // Create a LegCabin (Y) for the Leg LHR-BKK on BA's Inventory
    LegCabinKey lYLegCabinKey (lY);

    LegCabin& lLHRLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lLHRLeg, lLHRLegYCabin);
    FacBomManager::linkWithParent (lLHRLeg, lLHRLegYCabin);

    // Display the leg-cabin
    // STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

    // Create a LegCabin (Y) for the Leg BKK-SYD
    LegCabin& lBKKLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
    FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);
    // Display the leg-cabin
    // STDAIR_LOG_DEBUG ("LegCabin: " << lBKKLegYCabin.toString());

    // Step 0.7: fare family level
    // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
    const FamilyCode_T l1 ("EcoSaver");
    FareFamilyKey l1FareFamilyKey (l1);

    FareFamily& lLHRBKKSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
                                     lLHRBKKSegmentYCabin1Family);
    FacBomManager::linkWithParent (lLHRBKKSegmentYCabin,
                                   lLHRBKKSegmentYCabin1Family);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("FareFamily: "
    //                   << lLHRBKKSegmentYCabin1Family.toString());

    // Create a FareFamily (1)  for the Segment BKK-SYD, cabin Y on BA's Inv
    FareFamily& lBKKSYDSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                     lBKKSYDSegmentYCabin1Family);
    FacBomManager::linkWithParent (lBKKSYDSegmentYCabin,
                                   lBKKSYDSegmentYCabin1Family);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("FareFamily: "
```

```
    //                  << lLHRBKKSegmentYCabin1Family.toString());

    // Create a FareFamily (1)  for the Segment LHR-SYD, cabin Y on BA's Inv
    FareFamily& lLHRSYDSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                    lLHRSYDSegmentYCabin1Family);
    FacBomManager::linkWithParent (lLHRSYDSegmentYCabin,
                                   lLHRSYDSegmentYCabin1Family);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("FareFamily: "
    //                  << lLHRBKKSegmentYCabin1Family.toString());


    // Step 0.8: booking class level
    // Create a BookingClass (Q) for the Segment LHR-BKK, cabin Y,
    // fare family 1 on BA's Inv
    const ClassCode_T lQ ("Q");
    BookingClassKey lQBookingClassKey (lQ);

    BookingClass& lLHRBKKSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin1Family,
                                    lLHRBKKSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lLHRBKKSegmentYCabin1Family,
                                   lLHRBKKSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lLHRBKKSegmentYCabin,
                                    lLHRBKKSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lLHRBKKSegment,
                                    lLHRBKKSegmentYCabin1FamilyQClass);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                  << lLHRBKKSegmentYCabin1FamilyQClass.toString());

    // Create a BookingClass (Q) for the Segment BKK-SYD, cabin Y,
    // fare family 1 on BA's Inv
    BookingClass& lBKKSYDSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin1Family,
                                    lBKKSYDSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lBKKSYDSegmentYCabin1Family,
                                   lBKKSYDSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lBKKSYDSegmentYCabin,
                                    lBKKSYDSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lBKKSYDSegment,
                                    lBKKSYDSegmentYCabin1FamilyQClass);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                  << lLHRBKKSegmentYCabin1FamilyQClass.toString());

    // Create a BookingClass (Q) for the Segment LHR-SYD, cabin Y,
    // fare family 1 on BA's Inv
    BookingClass& lLHRSYDSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin1Family,
                                    lLHRSYDSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lLHRSYDSegmentYCabin1Family,
                                   lLHRSYDSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lLHRSYDSegmentYCabin,
                                    lLHRSYDSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lLHRSYDSegment,
```

```
                                    lLHRSYDSegmentYCabin1FamilyQClass);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lLHRBKKSegmentYCabin1FamilyQClass.toString());


    // ////// AF ///////
    // Step 0.2: Flight-date level
    // Create a FlightDate (AF084/20-MAR-2011) for AF's Inventory
    lFlightNumber = 84;
    lDate = Date_T (2011, 3, 20);
    lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

    FlightDate& lAF084_20110320_FD =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lAFInv, lAF084_20110320_FD);
    FacBomManager::linkWithParent (lAFInv, lAF084_20110320_FD);

    // Display the flight-date
    // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

    // Step 0.3: Segment-date level
    // Create a SegmentDate (CDG-SFO) for AF's Inventory
    const AirportCode_T lCDG ("CDG");
    const AirportCode_T lSFO ("SFO");
    const Duration_T l1040 (10, 40, 0);
    const Duration_T l1250 (12, 50, 0);
    const Duration_T l1110 (11, 10, 0);
    lSegmentDateKey = SegmentDateKey (lCDG, lSFO);

    SegmentDate& lCDGSFOSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGSFOSegment);
    FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGSFOSegment);

    // Add the routing leg key to the CDG-SFO segment.
    const std::string lAFCDGRoutingLegStr = "AF;84;2011-Mar-20;CDG";
    lCDGSFOSegment.addLegKey (lAFCDGRoutingLegStr);

    // Display the segment-date
    // STDAIR_LOG_DEBUG ("SegmentDate: " << lCDGSFOSegment.toString());

    // Fill the SegmentDate content
    lCDGSFOSegment.setBoardingDate (lDate);
    lCDGSFOSegment.setOffDate (lDate);
    lCDGSFOSegment.setBoardingTime (l1040);
    lCDGSFOSegment.setOffTime (l1250);
    lCDGSFOSegment.setElapsedTime (l1110);

    // Step 0.4: Leg-date level
    // Create a LegDate (CDG) for AF's Inventory
    lLegDateKey = LegDateKey (lCDG);

    LegDate& lCDGLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lAF084_20110320_FD, lCDGLeg);
    FacBomManager::linkWithParent (lAF084_20110320_FD, lCDGLeg);

    // Fill the LegDate content
    lCDGLeg.setOffPoint (lSFO);
    lCDGLeg.setBoardingDate (lDate);
    lCDGLeg.setOffDate (lDate);
    lCDGLeg.setBoardingTime (l1040);
    lCDGLeg.setOffTime (l1250);
    lCDGLeg.setElapsedTime (l1110);

    // Display the leg-date
```

```
    // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

    // Step 0.5: segment-cabin level
    // Create a SegmentCabin (Y) for the Segment CDG-SFO of AF's Inventory
    SegmentCabin& lCDGSFOSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lCDGSFOSegment, lCDGSFOSegmentYCabin);
    FacBomManager::linkWithParent (lCDGSFOSegment, lCDGSFOSegmentYCabin);

    // Display the segment-cabin
    // STDAIR_LOG_DEBUG ("SegmentCabin: " << lCDGSFOSegmentYCabin.toString());

    // Step 0.6: leg-cabin level
    // Create a LegCabin (Y) for the Leg CDG-SFO on AF's Inventory
    LegCabin& lCDGLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lCDGLeg, lCDGLegYCabin);
    FacBomManager::linkWithParent (lCDGLeg, lCDGLegYCabin);

    // Display the leg-cabin
    // STDAIR_LOG_DEBUG ("LegCabin: " << lLHRLegYCabin.toString());

    // Step 0.7: fare family level
    // Create a fareFamily (1) for the Segment CDG-SFO, cabin Y on AF's Inv
    FareFamily& lCDGSFOSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                    lCDGSFOSegmentYCabin1Family);
    FacBomManager::linkWithParent (lCDGSFOSegmentYCabin,
                                   lCDGSFOSegmentYCabin1Family);

    // Display the fare family
    // STDAIR_LOG_DEBUG ("fareFamily: "
    //
    //                   << lCDGSFOSegmentYCabin1Family.toString());

    // Step 0.8: booking class level Create a BookingClass (Q) for the
    // Segment CDG-SFO, cabin Y, fare family 1 on AF's Inv
    BookingClass& lCDGSFOSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin1Family,
                                    lCDGSFOSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lCDGSFOSegmentYCabin1Family,
                                   lCDGSFOSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lCDGSFOSegmentYCabin,
                                    lCDGSFOSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lCDGSFOSegment,
                                    lCDGSFOSegmentYCabin1FamilyQClass);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lCDGSFOSegmentYCabin1FamilyQClass.toString());

    /*==========================================================================
      ==========================================================================
      ==========================================================================*/
    // Schedule:
    // BA:
    // Step 1: flight period level
    // Create a flight period for BA9:
    const DoWStruct lDoWSrtuct ("1111111");
    const Date_T lBA9DateRangeStart (2010, boost::gregorian::Jun, 6);
    const Date_T lBA9DateRangeEnd (2010, boost::gregorian::Jun, 7);
```

```
    const DatePeriod_T lBA9DatePeriod (lBA9DateRangeStart, lBA9DateRangeEnd);
    const PeriodStruct lBA9PeriodStruct (lBA9DatePeriod, lDoWSrtuct);

    lFlightNumber = FlightNumber_T (9);

    FlightPeriodKey lBA9FlightPeriodKey (lFlightNumber, lBA9PeriodStruct);

    FlightPeriod& lBA9FlightPeriod =
      FacBom<FlightPeriod>::instance().create (lBA9FlightPeriodKey);
    FacBomManager::addToListAndMap (lBAInv, lBA9FlightPeriod);
    FacBomManager::linkWithParent (lBAInv, lBA9FlightPeriod);

    // Step 2: segment period level
    // Create a segment period for LHR-SYD:

    SegmentPeriodKey lLHRSYDSegmentPeriodKey (lLHR, lSYD);

    SegmentPeriod& lLHRSYDSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lLHRSYDSegmentPeriodKey);
    FacBomManager::addToListAndMap (lBA9FlightPeriod, lLHRSYDSegmentPeriod);
    FacBomManager::linkWithParent (lBA9FlightPeriod, lLHRSYDSegmentPeriod);

    lLHRSYDSegmentPeriod.setBoardingTime (l2135);
    lLHRSYDSegmentPeriod.setOffTime (l1540);
    lLHRSYDSegmentPeriod.setElapsedTime (l1105);
    ClassList_String_T lYM ("YM");
    lLHRSYDSegmentPeriod.addCabinBookingClassList (lY,lYM);

    // AF:
    // Step 1: flight period level
    // Create a flight period for AF84:
    const Date_T lAF84DateRangeStart (2011, boost::gregorian::Mar, 20);
    const Date_T lAF84DateRangeEnd (2011, boost::gregorian::Mar, 21);
    const DatePeriod_T lAF84DatePeriod (lAF84DateRangeStart, lAF84DateRangeEnd);
    const PeriodStruct lAF84PeriodStruct (lAF84DatePeriod, lDoWSrtuct);

    lFlightNumber = FlightNumber_T (84);

    FlightPeriodKey lAF84FlightPeriodKey (lFlightNumber, lAF84PeriodStruct);

    FlightPeriod& lAF84FlightPeriod =
      FacBom<FlightPeriod>::instance().create (lAF84FlightPeriodKey);
    FacBomManager::addToListAndMap (lAFInv, lAF84FlightPeriod);
    FacBomManager::linkWithParent (lAFInv, lAF84FlightPeriod);

    // Step 2: segment period level
    // Create a segment period for CDG-SFO:

    SegmentPeriodKey lCDGSFOSegmentPeriodKey (lCDG, lSFO);

    SegmentPeriod& lCDGSFOSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lCDGSFOSegmentPeriodKey);
    FacBomManager::addToListAndMap (lAF84FlightPeriod, lCDGSFOSegmentPeriod);
    FacBomManager::linkWithParent (lAF84FlightPeriod, lCDGSFOSegmentPeriod);

    lCDGSFOSegmentPeriod.setBoardingTime (l1040);
    lCDGSFOSegmentPeriod.setOffTime (l1250);
    lCDGSFOSegmentPeriod.setElapsedTime (l1110);
    lCDGSFOSegmentPeriod.addCabinBookingClassList (lY,lYM);

    /*==========================================================================
      ==========================================================================
      ==========================================================================*/
    // O&D
```

```
    // Create an O&D Date (BA;9,2010-Jun-06;LHR,SYD) for BA's Inventory
    OnDString_T  lBALHRSYDOnDStr = "BA;9,2010-Jun-06;LHR,SYD";
    OnDStringList_T lBAOnDStrList;
    lBAOnDStrList.push_back (lBALHRSYDOnDStr);

    OnDDateKey lBAOnDDateKey (lBAOnDStrList);
    OnDDate& lBA_LHRSYD_OnDDate =
      FacBom<OnDDate>::instance().create (lBAOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lBAInv, lBA_LHRSYD_OnDDate);
    FacBomManager::linkWithParent (lBAInv, lBA_LHRSYD_OnDDate);

    // Add the segment
    FacBomManager::addToListAndMap (lBA_LHRSYD_OnDDate, lLHRSYDSegment);

    // Add total forecast info for cabin Y.
    const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
    const WTP_T lWTP750 = 750.0;
    const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
    lBA_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

    // Create an O&D Date (AF;84,2011-Mar-21;CDG,SFO) for AF's Inventory
    OnDString_T lAFLHRSYDOnDStr = "AF;9,2011-Mar-20;CDG,SFO";
    OnDStringList_T lAFOnDStrList;
    lAFOnDStrList.push_back (lAFLHRSYDOnDStr);

    OnDDateKey lAFOnDDateKey (lAFOnDStrList);
    OnDDate& lAF_LHRSYD_OnDDate =
      FacBom<OnDDate>::instance().create (lAFOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lAFInv, lAF_LHRSYD_OnDDate);
    FacBomManager::linkWithParent (lAFInv, lAF_LHRSYD_OnDDate);

    // Add the segment
    FacBomManager::addToListAndMap (lAF_LHRSYD_OnDDate, lLHRSYDSegment);

    // Add total forecast info for cabin Y.
    lAF_LHRSYD_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::
  buildSampleInventoryScheduleForFareFamilies (BomRoot& ioBomRoot) {

    // Inventory
    // Step 0.1: Inventory level
    // Get the Inventory SQ (already built by construction)
    const InventoryKey lSQKey ("SQ");
    Inventory& lSQInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                         lSQKey.toString());

    // SQ
    // Step 0.2: Flight-date level
    // Create a FlightDate (SQ747/8-FEB-2010) for SQ's Inventory
    const FlightNumber_T lFlightNumber747 = 747;
    const Date_T lDate (2010, 2, 8);
    const FlightDateKey lFlightDateKey (lFlightNumber747, lDate);

    FlightDate& lSQ747_20100208_FD =
        FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lSQInv, lSQ747_20100208_FD);
    FacBomManager::linkWithParent (lSQInv, lSQ747_20100208_FD);

    // Display the flight-date
    // STDAIR_LOG_DEBUG ("FlightDate: " << lSQ747_20100208_FD.toString());

```

```
    // Step 0.3: Segment-date level
    // Create a SegmentDate (SIN-BKK) for SQ's Inventory
    const AirportCode_T lSIN ("SIN");
    const AirportCode_T lBKK ("BKK");
    const Duration_T l0635 (6, 35, 0);
    const Duration_T l0800 (8, 0, 0);
    const Duration_T l0225 (2, 25, 0);
    const SegmentDateKey lSegmentDateKey (lSIN, lBKK);

    SegmentDate& lSINBKKSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINBKKSegment);
    FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINBKKSegment);

    // Add the routing leg key to the SIN-BKK segment.
    const std::string lSQSINRoutingLegStr = "SQ;747;2010-Feb-8;SIN";
    lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

    // Fill the SegmentDate content
    lSINBKKSegment.setBoardingDate (lDate);
    lSINBKKSegment.setOffDate (lDate);
    lSINBKKSegment.setBoardingTime (l0635);
    lSINBKKSegment.setOffTime (l0800);
    lSINBKKSegment.setElapsedTime (l0225);

    // Display the segment-date
    // STDAIR_LOG_DEBUG ("SegmentDate: " << lSINBKKSegment);

    // Step 0.4: Leg-date level
    // Create a LegDate (SIN) for SQ's Inventory
    const LegDateKey lLegDateKey (lSIN);

    LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lSQ747_20100208_FD, lSINLeg);
    FacBomManager::linkWithParent (lSQ747_20100208_FD, lSINLeg);

    // Fill the LegDate content
    lSINLeg.setOffPoint (lBKK);
    lSINLeg.setBoardingDate (lDate);
    lSINLeg.setOffDate (lDate);
    lSINLeg.setBoardingTime (l0635);
    lSINLeg.setOffTime (l0800);
    lSINLeg.setElapsedTime (l0225);

    // Display the leg-date
    // STDAIR_LOG_DEBUG ("LegDate: " << lSINLeg.toString());

    // Step 0.5: segment-cabin level
    // Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
    const CabinCode_T lY ("Y");
    const SegmentCabinKey lYSegmentCabinKey (lY);
    SegmentCabin& lSINBKKSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
    FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);
    lSINBKKSegmentYCabin.activateFareFamily ();

    // Display the segment-cabin
    // STDAIR_LOG_DEBUG ("SegmentCabin: " << lSINBKKSegmentYCabin.toString());

    // Step 0.6: leg-cabin level
    // Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
    const LegCabinKey lYLegCabinKey (lY);
    LegCabin& lSINLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
    FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);
```

```

    // Display the leg-cabin
    // STDAIR_LOG_DEBUG ("LegCabin: " << lSINLegYCabin.toString());

    // Step 0.7: fare family level
    // Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
    const FamilyCode_T l1 ("1");
    const FareFamilyKey l1FareFamilyKey (l1);
    FareFamily& lSINBKKSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1Family);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                                   lSINBKKSegmentYCabin1Family);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("FareFamily: "
    //                   << lSINBKKSegmentYCabin1Family.toString());

    // Create a FareFamily (2) for the Segment SIN-BKK, cabin Y on SQ's Inv
    const FamilyCode_T l2 ("2");
    const FareFamilyKey l2FareFamilyKey (l2);
    FareFamily& lSINBKKSegmentYCabin2Family =
        FacBom<FareFamily>::instance().create (l2FareFamilyKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin2Family);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                                   lSINBKKSegmentYCabin2Family);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("FareFamily: "
    //                   << lSINBKKSegmentYCabin2Family.toString());

    // Step 0.8: booking class level
    // Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
    // fare family 2 on SQ's Inv
    const ClassCode_T lClassY ("Y");
    const BookingClassKey lYBookingClassKey (lClassY);
    BookingClass& lSINBKKSegmentYCabin2FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                    lSINBKKSegmentYCabin2FamilyYClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
                                   lSINBKKSegmentYCabin2FamilyYClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin2FamilyYClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin2FamilyYClass);
    lSINBKKSegmentYCabin2FamilyYClass.setYield(1200);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin2FamilyYClass.toString());

    // Create a BookingClass (B) for the Segment SIN-BKK, cabin Y,
    // fare family 2 on SQ's Inv
    const ClassCode_T lB ("B");
    const BookingClassKey lBBookingClassKey (lB);
    BookingClass& lSINBKKSegmentYCabin2FamilyBClass =
      FacBom<BookingClass>::instance().create (lBBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin2Family,
                                    lSINBKKSegmentYCabin2FamilyBClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin2Family,
                                   lSINBKKSegmentYCabin2FamilyBClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
```

```
                                    lSINBKKSegmentYCabin2FamilyBClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin2FamilyBClass);
    lSINBKKSegmentYCabin2FamilyBClass.setYield(800);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin2FamilyBClass.toString());

    // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    const ClassCode_T lM ("M");
    const BookingClassKey lMBookingClassKey (lM);
    BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    lSINBKKSegmentYCabin1FamilyMClass.setYield(900);

    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin1FamilyMClass.toString());

    // Create a BookingClass (Q) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    const ClassCode_T lQ ("Q");
    const BookingClassKey lQBookingClassKey (lQ);
    BookingClass& lSINBKKSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin1FamilyQClass);
    lSINBKKSegmentYCabin1FamilyQClass.setYield(600);


    // Display the booking class
    // STDAIR_LOG_DEBUG ("BookingClass: "
    //                   << lSINBKKSegmentYCabin1FamilyQClass.toString());

    /*==========================================================================
      ==========================================================================
      ==========================================================================*/
    // Schedule:
    // SQ:
    // Step 1: flight period level
    // Create a flight period for SQ747:
    const DoWStruct lDoWSrtuct ("1111111");
    const Date_T lSQ747DateRangeStart (2010, boost::gregorian::Feb, 8);
    const Date_T lSQ747DateRangeEnd (2010, boost::gregorian::Feb, 9);
    const DatePeriod_T lSQ747DatePeriod (lSQ747DateRangeStart,
                                         lSQ747DateRangeEnd);
```

```
    const PeriodStruct lSQ747PeriodStruct (lSQ747DatePeriod, lDoWSrtuct);

    const FlightPeriodKey lSQ747FlightPeriodKey (lFlightNumber747,
                                                 lSQ747PeriodStruct);
    FlightPeriod& lSQ747FlightPeriod =
      FacBom<FlightPeriod>::instance().create (lSQ747FlightPeriodKey);
    FacBomManager::addToListAndMap (lSQInv, lSQ747FlightPeriod);
    FacBomManager::linkWithParent (lSQInv, lSQ747FlightPeriod);

    // Step 2: segment period level
    // Create a segment period for SIN-BKK:

    const SegmentPeriodKey lSINBKKSegmentPeriodKey (lSIN, lBKK);
    SegmentPeriod& lSINBKKSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lSINBKKSegmentPeriodKey);
    FacBomManager::addToListAndMap (lSQ747FlightPeriod, lSINBKKSegmentPeriod);
    FacBomManager::linkWithParent (lSQ747FlightPeriod, lSINBKKSegmentPeriod);

    ClassList_String_T lYBMQ  ("YBMQ");
    lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYBMQ);
    lSINBKKSegmentPeriod.setBoardingTime (l0635);
    lSINBKKSegmentPeriod.setOffTime (l0800);
    lSINBKKSegmentPeriod.setElapsedTime (l0225);

    /*==========================================================================
      ==========================================================================
      ==========================================================================*/
    // O&D
    // Create an O&D Date (SQ;747,2011-Feb-14;SIN,BKK) for SQ's Inventory
    const OnDString_T lSQSINBKKOnDStr = "SQ;747,2011-Feb-14;SIN,BKK";
    OnDStringList_T lSQOnDStrList;
    lSQOnDStrList.push_back (lSQSINBKKOnDStr);

    const OnDDateKey lSQOnDDateKey (lSQOnDStrList);
    OnDDate& lSQ_SINBKK_OnDDate =
      FacBom<OnDDate>::instance().create (lSQOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
    FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);
    // Add total forecast info for cabin Y.
    const MeanStdDevPair_T lMean120StdDev12 (120.0, 12.0);
    const WTP_T lWTP1000 = 1000.0;
    const WTPDemandPair_T lWTP1000Mean120StdDev12 (lWTP1000, lMean120StdDev12);
    lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP1000Mean120StdDev12);

    // Add the segment
    FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);
  }

  // ////////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildDummyLegSegmentAccesses (BomRoot& ioBomRoot) {

    /* Build the direct accesses between the dummy segment cabins and the dummy
     * leg cabins within the dummy flight dates (the dummy fare family
     * flight date and the classic dummy flight date).
     *
     * As for now (May 2012), that method is called only by RMOL.
     * It is a substitute for the code doing it automatically located in AirInv.
     * See the AIRINV::InventoryManager::createDirectAccesses command.
     */

    // ////// Dummy Inventory Leg Segment Accesses ///////
    // Retrieve the (sample) segment-cabin.
    SegmentCabin& lDummySegmentCabin =
```

```
      BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

    // Retrieve the (sample) leg-cabin.
    LegCabin& lDummyLegCabin =
      BomRetriever::retrieveDummyLegCabin (ioBomRoot);

    // Links between the segment-date and the leg-date
    FacBomManager::addToListAndMap (lDummyLegCabin, lDummySegmentCabin);
    FacBomManager::addToListAndMap (lDummySegmentCabin, lDummyLegCabin);

    // ////// Fare Families Dummy Inventory Leg Segment Accesses ///////
    const bool isForFareFamilies = true;
    // Retrieve the (sample) segment-cabin for fare families.
    SegmentCabin& lFFDummySegmentCabin =
      BomRetriever::retrieveDummySegmentCabin (ioBomRoot, isForFareFamilies);

    // Retrieve the (sample) leg-cabin for fare families.
    stdair::LegCabin& lFFDummyLegCabin =
      stdair::BomRetriever::retrieveDummyLegCabin (ioBomRoot,
                                                   isForFareFamilies);

    // Links between the segment-date and the leg-date for fare families.
    FacBomManager::addToListAndMap (lFFDummyLegCabin, lFFDummySegmentCabin);
    FacBomManager::addToListAndMap (lFFDummySegmentCabin, lFFDummyLegCabin);
  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildCompleteDummyInventory  (BomRoot& ioBomRoot) {

    // Build a dummy inventory, containing a dummy flight-date with a
    // single segment-cabin and a single leg-cabin.
    const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
    buildDummyInventory (ioBomRoot, lCapacity);

    // Retrieve the (sample) segment-cabin.
    SegmentCabin& lDummySegmentCabin =
      BomRetriever::retrieveDummySegmentCabin (ioBomRoot);

    // Retrieve the (sample) leg-cabin.
    LegCabin& lDummyLegCabin =
      BomRetriever::retrieveDummyLegCabin (ioBomRoot);

    // Add some booking classes to the dummy segment-cabin and some
    // virtual ones to the dummy leg-cabin.
    // First booking class yield and demand information.
    Yield_T lYield = 100;
    MeanValue_T lMean = 20;
    StdDevValue_T lStdDev= 9;
    BookingClassKey lBCKey (DEFAULT_CLASS_CODE);

    BookingClass& lDummyBookingClass =
      FacBom<BookingClass>::instance().create (lBCKey);
    lDummyBookingClass.setYield (lYield);
    lDummyBookingClass.setMean (lMean);
    lDummyBookingClass.setStdDev (lStdDev);
    // Add a booking class to the segment-cabin.
    FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);
    BookingClassList_T lDummyBookingClassList;
    lDummyBookingClassList.push_back(&lDummyBookingClass);

    VirtualClassStruct lDummyVirtualClass (lDummyBookingClassList);
    lDummyVirtualClass.setYield (lYield);
    lDummyVirtualClass.setMean (lMean);
    lDummyVirtualClass.setStdDev (lStdDev);
    // Add the corresponding virtual class to the leg-cabin.
    lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

```

```
    // Second booking class yield and demand information.
    lYield = 70;
    lMean = 45;
    lStdDev= 12;
    lDummyBookingClass.setYield (lYield);
    lDummyBookingClass.setMean (lMean);
    lDummyBookingClass.setStdDev (lStdDev);
    // Add a booking class to the segment-cabin.
    FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

    lDummyVirtualClass.setYield (lYield);
    lDummyVirtualClass.setMean (lMean);
    lDummyVirtualClass.setStdDev (lStdDev);
    // Add the corresponding virtual class to the leg-cabin.
    lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

    // Third booking class yield and demand information.
    lYield = 42;
    lMean = 80;
    lStdDev= 16;
    lDummyBookingClass.setYield (lYield);
    lDummyBookingClass.setMean (lMean);
    lDummyBookingClass.setStdDev (lStdDev);
    // Add a booking class to the segment-cabin.
    FacBomManager::addToList (lDummySegmentCabin, lDummyBookingClass);

    lDummyVirtualClass.setYield (lYield);
    lDummyVirtualClass.setMean (lMean);
    lDummyVirtualClass.setStdDev (lStdDev);
    // Add the corresponding virtual class to the leg-cabin.
    lDummyLegCabin.addVirtualClass (lDummyVirtualClass);

  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildDummyInventory (BomRoot& ioBomRoot,
                                           const CabinCapacity_T& iCapacity) {
    // Inventory
    const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
    Inventory& lInv = FacBom<Inventory>::instance().create (lInventoryKey);
    FacBomManager::addToListAndMap (ioBomRoot, lInv);
    FacBomManager::linkWithParent (ioBomRoot, lInv);

    // Add the airline feature object to the dummy inventory
    const AirlineFeatureKey lAirlineFeatureKey (DEFAULT_AIRLINE_CODE);
    AirlineFeature& lAirlineFeature =
      FacBom<AirlineFeature>::instance().create (lAirlineFeatureKey);
    FacBomManager::setAirlineFeature (lInv, lAirlineFeature);
    FacBomManager::linkWithParent (lInv, lAirlineFeature);
    // Link the airline feature object with the top of the BOM tree
    FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeature);

    // Flight-date
    FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER, DEFAULT_DEPARTURE_DATE);
    FlightDate& lFlightDate =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lInv, lFlightDate);
    FacBomManager::linkWithParent (lInv, lFlightDate);

    // Leg-date
    LegDateKey lLegDateKey (DEFAULT_ORIGIN);
    LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lFlightDate, lLeg);
    FacBomManager::linkWithParent (lFlightDate, lLeg);

    // Fill the LegDate content
    lLeg.setOffPoint (DEFAULT_DESTINATION);
```

```
    lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setBoardingTime (Duration_T (14, 0, 0));
    lLeg.setOffTime (Duration_T (16, 0, 0));
    lLeg.setElapsedTime (Duration_T (8, 0, 0));

    // Leg-cabin
    LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
    LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
    FacBomManager::addToListAndMap (lLeg, lLegCabin);
    FacBomManager::linkWithParent (lLeg, lLegCabin);

    lLegCabin.setCapacities (iCapacity);
    lLegCabin.setAvailabilityPool (iCapacity);

    // Segment-date
    SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
    SegmentDate& lSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lFlightDate, lSegment);
    FacBomManager::linkWithParent (lFlightDate, lSegment);

    // Add the routing leg key to the dummy segment.
    std::ostringstream oStr;
    oStr << DEFAULT_AIRLINE_CODE << ";"
         << DEFAULT_FLIGHT_NUMBER << ";"
         << DEFAULT_DEPARTURE_DATE << ";"
         << DEFAULT_ORIGIN;
    lSegment.addLegKey (oStr.str());

    // Fill the SegmentDate content
    lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setBoardingTime (Duration_T (14, 0, 0));
    lSegment.setOffTime (Duration_T (16, 0, 0));
    lSegment.setElapsedTime (Duration_T (8, 0, 0));

    // Segment-cabin
    SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
    SegmentCabin& lSegmentCabin =
      FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
    FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
    FacBomManager::linkWithParent (lSegment, lSegmentCabin);

    // Create a FareFamily (1) for the Segment LHR-BKK, cabin Y on BA's Inv
    const FamilyCode_T l1 ("EcoSaver");
    FareFamilyKey l1FareFamilyKey (l1);

    FareFamily& lSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
    FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);

    // Create a booking-class
    const ClassCode_T lQ ("Q");
    BookingClassKey lQBookingClassKey (lQ);

    BookingClass& lSegmentYCabin1FamilyQClass =
      FacBom<BookingClass>::instance().create (lQBookingClassKey);
    FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                    lSegmentYCabin1FamilyQClass);
    FacBomManager::linkWithParent (lSegmentYCabin1Family,
                                   lSegmentYCabin1FamilyQClass);

    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyQClass);
    FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyQClass);

```

```
    /*==========================================================================
      ==========================================================================
      ==========================================================================*/
    // Schedule:
    // XX:
    // Step 1: flight period level
    // Create a flight period for XX:
    const DoWStruct lDoWSrtuct ("1111111");
    const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
    const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
    const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
    const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);

    FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER, lXXPeriodStruct);

    FlightPeriod& lXXFlightPeriod =
      FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
    FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
    FacBomManager::linkWithParent (lInv, lXXFlightPeriod);

    // Step 2: segment period level
    // Create a segment period

    SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);

    SegmentPeriod& lXXSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
    FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
    FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);

    lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
    lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
    lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
    const CabinCode_T lY ("Y");
    const ClassList_String_T lYQ ("YQ");
    lXXSegmentPeriod.addCabinBookingClassList (lY,lYQ);


  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::
  buildCompleteDummyInventoryForFareFamilies (BomRoot& ioBomRoot) {

    // Build a dummy inventory, containing a dummy flight-date with a
    // single segment-cabin and a single leg-cabin (for fare families
    // algorithms)

    // Get the default Inventory object (already built in by construction)
    const InventoryKey lInventoryKey (DEFAULT_AIRLINE_CODE);
    Inventory& lInv = BomManager::getObject<Inventory>(ioBomRoot,
                                                       lInventoryKey.toString());

    // Create a dummy Flight-date
    const FlightDateKey lFlightDateKey(DEFAULT_FLIGHT_NUMBER_FF,
                                       DEFAULT_DEPARTURE_DATE);
    FlightDate& lFlightDate =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lInv, lFlightDate);
    FacBomManager::linkWithParent (lInv, lFlightDate);

    // Create a dummy Leg-date
    LegDateKey lLegDateKey (DEFAULT_ORIGIN);
    LegDate& lLeg = FacBom<LegDate>::instance().create (lLegDateKey);
```

```
    FacBomManager::addToListAndMap (lFlightDate, lLeg);
    FacBomManager::linkWithParent (lFlightDate, lLeg);

    // Fill the LegDate content
    lLeg.setOffPoint (DEFAULT_DESTINATION);
    lLeg.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setOffDate (DEFAULT_DEPARTURE_DATE);
    lLeg.setBoardingTime (Duration_T (14, 0, 0));
    lLeg.setOffTime (Duration_T (16, 0, 0));
    lLeg.setElapsedTime (Duration_T (8, 0, 0));

    // Create a dummy Leg-cabin
    const LegCabinKey lLegCabinKey (DEFAULT_CABIN_CODE);
    LegCabin& lLegCabin = FacBom<LegCabin>::instance().create (lLegCabinKey);
    FacBomManager::addToListAndMap (lLeg, lLegCabin);
    FacBomManager::linkWithParent (lLeg, lLegCabin);
    const CabinCapacity_T lCapacity = DEFAULT_CABIN_CAPACITY;
    lLegCabin.setCapacities (lCapacity);
    lLegCabin.setAvailabilityPool (lCapacity);

    // Create a dummy Segment-date
    const SegmentDateKey lSegmentDateKey (DEFAULT_ORIGIN, DEFAULT_DESTINATION);
    SegmentDate& lSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lFlightDate, lSegment);
    FacBomManager::linkWithParent (lFlightDate, lSegment);

    // Add the routing leg key to the dummy segment.
    std::ostringstream oStr;
    oStr << DEFAULT_AIRLINE_CODE << ";"
         << DEFAULT_FLIGHT_NUMBER << ";"
         << DEFAULT_DEPARTURE_DATE << ";"
         << DEFAULT_ORIGIN;
    lSegment.addLegKey (oStr.str());

    // Fill the SegmentDate content
    lSegment.setBoardingDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setOffDate (DEFAULT_DEPARTURE_DATE);
    lSegment.setBoardingTime (Duration_T (14, 0, 0));
    lSegment.setOffTime (Duration_T (16, 0, 0));
    lSegment.setElapsedTime (Duration_T (8, 0, 0));

    // Create a dummy Segment-cabin
    const SegmentCabinKey lSegmentCabinKey (DEFAULT_CABIN_CODE);
    SegmentCabin& lSegmentCabin =
      FacBom<SegmentCabin>::instance().create (lSegmentCabinKey);
    FacBomManager::addToListAndMap (lSegment, lSegmentCabin);
    FacBomManager::linkWithParent (lSegment, lSegmentCabin);

    // Create a dummy FareFamily (FF1)
    const FamilyCode_T l1 ("FF1");
    const FareFamilyKey l1FareFamilyKey (l1);

    FareFamily& lSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    // Set the forecasted demand
    // TODO change the size (hard code)
    MeanStdDevPairVector_T lDemandVector1FareFamily;
    const unsigned int size = 16;
    for (unsigned int idx = 0; idx < size; ++idx) {
      double i = static_cast<double> (idx);
      MeanStdDevPair_T lMeanStdDevPair (i/4.0, i/20.0);
      lDemandVector1FareFamily.push_back(lMeanStdDevPair);
    }
    lSegmentYCabin1Family.setMeanStdDev(lDemandVector1FareFamily);
    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1Family);
    FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin1Family);
```

```

    // Create a dummy booking-class
    const ClassCode_T lY ("Y");
    const BookingClassKey lYBookingClassKey (lY);

    BookingClass& lSegmentYCabin1FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    Yield_T lYield = 1000;
    lSegmentYCabin1FamilyYClass.setYield(lYield);
    FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                    lSegmentYCabin1FamilyYClass);
    FacBomManager::linkWithParent (lSegmentYCabin1Family,
                                   lSegmentYCabin1FamilyYClass);

    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyYClass);
    FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyYClass);

    // Create a second dummy booking-class
    const ClassCode_T lU ("U");
    const BookingClassKey lUBookingClassKey (lU);

    BookingClass& lSegmentYCabin1FamilyUClass =
      FacBom<BookingClass>::instance().create (lUBookingClassKey);
    lYield = 600;
    lSegmentYCabin1FamilyUClass.setYield(lYield);
    FacBomManager::addToListAndMap (lSegmentYCabin1Family,
                                    lSegmentYCabin1FamilyUClass);
    FacBomManager::linkWithParent (lSegmentYCabin1Family,
                                   lSegmentYCabin1FamilyUClass);

    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin1FamilyUClass);
    FacBomManager::addToListAndMap (lSegment, lSegmentYCabin1FamilyUClass);

    // Create a second dummy FareFamily (2)
    const FamilyCode_T l2 ("FF2");
    const FareFamilyKey l2FareFamilyKey (l2);

    FareFamily& lSegmentYCabin2Family =
      FacBom<FareFamily>::instance().create (l2FareFamilyKey);
    // Set the forecasted demand
    // TODO change the size (hard code)
    MeanStdDevPairVector_T lDemandVector2FareFamily;
    for (unsigned int idx = 0; idx < size; ++idx) {
      double i = static_cast<double> (idx);
      MeanStdDevPair_T lMeanStdDevPair (i/2.0, i/10.0);
      lDemandVector2FareFamily.push_back(lMeanStdDevPair);
    }
    lSegmentYCabin2Family.setMeanStdDev(lDemandVector2FareFamily);

    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2Family);
    FacBomManager::linkWithParent (lSegmentCabin, lSegmentYCabin2Family);

    // Create a third dummy booking-class
    const ClassCode_T lO ("O");
    const BookingClassKey lOBookingClassKey (lO);

    BookingClass& lSegmentYCabin2FamilyOClass =
      FacBom<BookingClass>::instance().create (lOBookingClassKey);
    lYield = 750;
    lSegmentYCabin2FamilyOClass.setYield(lYield);
    FacBomManager::addToListAndMap (lSegmentYCabin2Family,
                                    lSegmentYCabin2FamilyOClass);
    FacBomManager::linkWithParent (lSegmentYCabin2Family,
                                   lSegmentYCabin2FamilyOClass);

    FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyOClass);
    FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyOClass);
```

```
01324
01325      // Create a fourth dummy booking-class
01326     const ClassCode_T lQ ("Q");
01327     const BookingClassKey lQBookingClassKey (lQ);
01328
01329     BookingClass& lSegmentYCabin2FamilyQClass =
01330       FacBom<BookingClass>::instance().create (lQBookingClassKey);
01331     lYield = 400;
01332     lSegmentYCabin2FamilyQClass.setYield(lYield);
01333     FacBomManager::addToListAndMap (lSegmentYCabin2Family,
01334                                     lSegmentYCabin2FamilyQClass);
01335     FacBomManager::linkWithParent (lSegmentYCabin2Family,
01336                                    lSegmentYCabin2FamilyQClass);
01337
01338     FacBomManager::addToListAndMap (lSegmentCabin, lSegmentYCabin2FamilyQClass);
01339     FacBomManager::addToListAndMap (lSegment, lSegmentYCabin2FamilyQClass);
01340
01341
01342     /*==========================================================================
      =====
01343       ==========================================================================
      =====
01344       ==========================================================================
      =====*/
01345     // Schedule:
01346     // XX:
01347     // Step 1: flight period level
01348     // Create a flight period for XX:
01349     const DoWStruct lDoWSrtuct ("1111111");
01350     const Date_T lXXDateRangeStart (DEFAULT_DEPARTURE_DATE);
01351     const Date_T lXXDateRangeEnd (DEFAULT_DEPARTURE_DATE);
01352     const DatePeriod_T lXXDatePeriod (lXXDateRangeStart, lXXDateRangeEnd);
01353     const PeriodStruct lXXPeriodStruct (lXXDatePeriod, lDoWSrtuct);
01354
01355     const FlightPeriodKey lXXFlightPeriodKey (DEFAULT_FLIGHT_NUMBER_FF,
01356                                               lXXPeriodStruct);
01357
01358     FlightPeriod& lXXFlightPeriod =
01359       FacBom<FlightPeriod>::instance().create (lXXFlightPeriodKey);
01360     FacBomManager::addToListAndMap (lInv, lXXFlightPeriod);
01361     FacBomManager::linkWithParent (lInv, lXXFlightPeriod);
01362
01363     // Step 2: segment period level
01364     // Create a segment period
01365     const SegmentPeriodKey lXXSegmentPeriodKey (DEFAULT_ORIGIN,
01366                                                 DEFAULT_DESTINATION);
01367
01368     SegmentPeriod& lXXSegmentPeriod =
01369       FacBom<SegmentPeriod>::instance().create (lXXSegmentPeriodKey);
01370     FacBomManager::addToListAndMap (lXXFlightPeriod, lXXSegmentPeriod);
01371     FacBomManager::linkWithParent (lXXFlightPeriod, lXXSegmentPeriod);
01372
01373     lXXSegmentPeriod.setBoardingTime (Duration_T (14, 0, 0));
01374     lXXSegmentPeriod.setOffTime (Duration_T (16, 0, 0));
01375     lXXSegmentPeriod.setElapsedTime (Duration_T (8, 0, 0));
01376     const CabinCode_T lYCabin ("Y");
01377     const ClassList_String_T lYUOQ ("YUOQ");
01378     lXXSegmentPeriod.addCabinBookingClassList (lYCabin,lYUOQ);
01379
01380   }
01381
01382   // ////////////////////////////////////////////////////////////////////
01383   void CmdBomManager::buildSamplePricing (BomRoot& ioBomRoot) {
01384
01385     // Set the airport-pair primary key.
01386     const AirportPairKey lAirportPairKey (AIRPORT_LHR, AIRPORT_SYD);
01387
```

```
    // Create the AirportPairKey object and link it to the BOM tree root.
    AirportPair& lAirportPair =
      FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lAirportPair);

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2011, boost::gregorian::Jan, 15);
    const Date_T lDateRangeEnd (2011, boost::gregorian::Dec, 31);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
    FacBomManager::linkWithParent (lAirportPair, lDatePeriod);

    // Set the point-of-sale-channel primary key.
    const PosChannelKey lPosChannelKey (POS_LHR, CHANNEL_DN);

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
    FacBomManager::linkWithParent (lDatePeriod, lPosChannel);

    // Set the fare time-period primary key.
    const Time_T lTimeRangeStart (0, 0, 0);
    const Time_T lTimeRangeEnd (23, 0, 0);
    const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lTimePeriod =
      FacBom<TimePeriod>::instance().create (lTimePeriodKey);
    FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
    FacBomManager::linkWithParent (lPosChannel, lTimePeriod);

    // Pricing -- Generate the FareRule
    const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                            NO_ADVANCE_PURCHASE,
                                            SATURDAY_STAY,
                                            CHANGE_FEES,
                                            NON_REFUNDABLE,
                                            NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lFareFeatures =
      FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
    FacBomManager::addToListAndMap (lTimePeriod, lFareFeatures);
    FacBomManager::linkWithParent (lTimePeriod, lFareFeatures);

    // Revenue Accounting -- Generate the YieldRule
    const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ROUND_TRIP,
                                              CABIN_Y);

    // Create the YieldFeaturesKey and link it to the TimePeriod object.
    YieldFeatures& lYieldFeatures =
      FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
    FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
    FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);

    // Generate Segment Features and link them to their respective
    // fare and yield rules.
    AirlineCodeList_T lAirlineCodeList;
    lAirlineCodeList.push_back (AIRLINE_CODE_BA);
    ClassList_StringList_T lClassCodeList;
```

```
    lClassCodeList.push_back (CLASS_CODE_Y);
    const AirlineClassListKey lAirlineClassListKey (lAirlineCodeList,
                                                    lClassCodeList);

    // Create the AirlineClassList
    AirlineClassList& lAirlineClassList =
      FacBom<AirlineClassList>::instance().create (lAirlineClassListKey);
    // Link the AirlineClassList to the FareFeatures object
    lAirlineClassList.setFare (900);
    FacBomManager::addToListAndMap (lFareFeatures, lAirlineClassList);
    FacBomManager::linkWithParent (lFareFeatures, lAirlineClassList);

    // Link the AirlineClassList to the YieldFeatures object
    lAirlineClassList.setYield (900);
    FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassList);
    // \todo (gsabatier): the following calls overrides the parent for
    //       lAirlineClassList. Check that it is what is actually wanted.
    FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassList);
  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildSamplePricingForFareFamilies (BomRoot& ioBomRoot) {

    // Get the airport-pair primary key SIN-BKK
    // (already built by construction)
    const AirportPairKey lAirportPairKey ("SIN", "BKK");
    AirportPair& lAirportPair =
      BomManager::getObject<AirportPair>(ioBomRoot, lAirportPairKey.toString());

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2010, boost::gregorian::Feb, 1);
    const Date_T lDateRangeEnd (2011, boost::gregorian::Feb, 15);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lAirportPair, lDatePeriod);
    FacBomManager::linkWithParent (lAirportPair, lDatePeriod);

    // Set the point-of-sale-channel primary key.
    const PosChannelKey lPosChannelKey ("SIN", CHANNEL_IN);

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lDatePeriod, lPosChannel);
    FacBomManager::linkWithParent (lDatePeriod, lPosChannel);

    // Set the fare time-period primary key.
    const Time_T lTimeRangeStart (0, 0, 0);
    const Time_T lTimeRangeEnd (23, 0, 0);
    const TimePeriodKey lTimePeriodKey (lTimeRangeStart, lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lTimePeriod =
      FacBom<TimePeriod>::instance().create (lTimePeriodKey);
    FacBomManager::addToListAndMap (lPosChannel, lTimePeriod);
    FacBomManager::linkWithParent (lPosChannel, lTimePeriod);

    // Pricing -- Generate the FareRule
    const DayDuration_T ONE_MONTH_ADVANCE_PURCHASE = 30;
    // Generate the first FareFeatures for the class Q
    const FareFeaturesKey lFareFeaturesQKey (TRIP_TYPE_ONE_WAY,
                                             ONE_MONTH_ADVANCE_PURCHASE,
                                             SATURDAY_STAY,
```

```
                                          CHANGE_FEES,
                                          NON_REFUNDABLE,
                                          NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lFareFeaturesQ =
      FacBom<FareFeatures>::instance().create (lFareFeaturesQKey);
    FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesQ);
    FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesQ);

    // Generate the second FareFeatures for the class M
    const FareFeaturesKey lFareFeaturesMKey (TRIP_TYPE_ONE_WAY,
                                          NO_ADVANCE_PURCHASE,
                                          SATURDAY_STAY,
                                          CHANGE_FEES,
                                          NON_REFUNDABLE,
                                          NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lFareFeaturesM =
      FacBom<FareFeatures>::instance().create (lFareFeaturesMKey);
    FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesM);
    FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesM);

    // Generate the third FareFeatures for the class B
    const FareFeaturesKey lFareFeaturesBKey (TRIP_TYPE_ONE_WAY,
                                           ONE_MONTH_ADVANCE_PURCHASE,
                                           SATURDAY_STAY,
                                           NO_CHANGE_FEES,
                                           NO_NON_REFUNDABLE,//Refundable
                                           NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lFareFeaturesB =
      FacBom<FareFeatures>::instance().create (lFareFeaturesBKey);
    FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesB);
    FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesB);

    // Generate the fourth FareFeatures for the class Y
    const FareFeaturesKey lFareFeaturesYKey (TRIP_TYPE_ONE_WAY,
                                          NO_ADVANCE_PURCHASE,
                                          SATURDAY_STAY,
                                          NO_CHANGE_FEES,
                                          NO_NON_REFUNDABLE,//Refundable
                                          NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lFareFeaturesY =
      FacBom<FareFeatures>::instance().create (lFareFeaturesYKey);
    FacBomManager::addToListAndMap (lTimePeriod, lFareFeaturesY);
    FacBomManager::linkWithParent (lTimePeriod, lFareFeaturesY);

    // Revenue Accounting -- Generate the YieldRule
    const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
                                            CABIN_Y);

    // Create the YieldFeaturesKey and link it to the TimePeriod object.
    YieldFeatures& lYieldFeatures =
      FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
    FacBomManager::addToListAndMap (lTimePeriod, lYieldFeatures);
    FacBomManager::linkWithParent (lTimePeriod, lYieldFeatures);

    // Generate Segment Features and link them to their respective
    // fare and yield rules.
    AirlineCodeList_T lAirlineCodeList;
    lAirlineCodeList.push_back ("SQ");

```

```
    ClassList_StringList_T lClassYList;
    lClassYList.push_back (CLASS_CODE_Y);
    const AirlineClassListKey lAirlineClassYListKey (lAirlineCodeList,
                                                     lClassYList);

    // Create the AirlineClassList
    AirlineClassList& lAirlineClassYList =
        FacBom<AirlineClassList>::instance().create (lAirlineClassYListKey);
    // Link the AirlineClassList to the FareFeatures object
    FacBomManager::addToListAndMap (lFareFeaturesY, lAirlineClassYList);
    FacBomManager::linkWithParent (lFareFeaturesY, lAirlineClassYList);
    lAirlineClassYList.setFare (1200);
    lAirlineClassYList.setYield (1200);

    // Link the AirlineClassList to the YieldFeatures object
    FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassYList);
    // \todo (gsabatier): the following calls overrides the parent for
    //       lAirlineClassList. Check that it is what is actually wanted.
    FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassYList);

    ClassList_StringList_T lClassBList;
    lClassBList.push_back ("B");
    const AirlineClassListKey lAirlineClassBListKey (lAirlineCodeList,
                                                      lClassBList);
    // Create the AirlineClassList
    AirlineClassList& lAirlineClassBList =
      FacBom<AirlineClassList>::instance().create (lAirlineClassBListKey);
    // Link the AirlineClassList to the FareFeatures object
    FacBomManager::addToListAndMap (lFareFeaturesB, lAirlineClassBList);
    FacBomManager::linkWithParent (lFareFeaturesB, lAirlineClassBList);
    lAirlineClassBList.setFare (800);
    lAirlineClassBList.setYield (800);

    // Link the AirlineClassList to the YieldFeatures object
    FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassBList);
    // \todo (gsabatier): the following calls overrides the parent for
    //       lAirlineClassList. Check that it is what is actually wanted.
    FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassBList);

    ClassList_StringList_T lClassMList;
    lClassMList.push_back ("M");
    const AirlineClassListKey lAirlineClassMListKey (lAirlineCodeList,
                                                     lClassMList);

    // Create the AirlineClassList
    AirlineClassList& lAirlineClassMList =
      FacBom<AirlineClassList>::instance().create (lAirlineClassMListKey);
    // Link the AirlineClassList to the FareFeatures object
    FacBomManager::addToListAndMap (lFareFeaturesM, lAirlineClassMList);
    FacBomManager::linkWithParent (lFareFeaturesM, lAirlineClassMList);
    lAirlineClassMList.setFare (900);
    lAirlineClassMList.setYield (900);

    // Link the AirlineClassList to the YieldFeatures object
    FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassMList);
    // \todo (gsabatier): the following calls overrides the parent for
    //       lAirlineClassList. Check that it is what is actually wanted.
    FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassMList);

    ClassList_StringList_T lClassQList;
    lClassQList.push_back ("Q");
    const AirlineClassListKey lAirlineClassQListKey (lAirlineCodeList,
                                                      lClassQList);

    // Create the AirlineClassList
    AirlineClassList& lAirlineClassQList =
      FacBom<AirlineClassList>::instance().create (lAirlineClassQListKey);
```

```
    // Link the AirlineClassList to the FareFeatures object
    FacBomManager::addToListAndMap (lFareFeaturesQ, lAirlineClassQList);
    FacBomManager::linkWithParent (lFareFeaturesQ, lAirlineClassQList);
    lAirlineClassQList.setFare (600);
    lAirlineClassQList.setYield (600);

    // Link the AirlineClassList to the YieldFeatures object
    FacBomManager::addToListAndMap (lYieldFeatures, lAirlineClassQList);
    // \todo (gsabatier): the following calls overrides the parent for
    //       lAirlineClassList. Check that it is what is actually wanted.
    FacBomManager::linkWithParent (lYieldFeatures, lAirlineClassQList);

  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::
  buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
) {

    // Clean the list
    ioTravelSolutionList.clear();

    //
    const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");

    // Add the segment date key to the travel solution
    TravelSolutionStruct lTS;
    lTS.addSegment (lBA9_SegmentDateKey);

    // Add the travel solution to the list
    ioTravelSolutionList.push_back (lTS);
  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::
  buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {

    // Clean the list
    ioTravelSolutionList.clear();

    //
    const std::string lBA9_SegmentDateKey ("BA, 9, 2011-06-10, LHR, SYD, 21:45");

    // Add the segment date key to the travel solution
    TravelSolutionStruct lTS1;
    lTS1.addSegment (lBA9_SegmentDateKey);

    // Fare option number 1
    const ClassCode_T lClassPathQ (CLASS_CODE_Q);
    const Fare_T lFare900 (900);
    const ChangeFees_T lChangeFee (CHANGE_FEES);
    const NonRefundable_T isNonRefundable (NON_REFUNDABLE);
    const SaturdayStay_T lSaturdayStay (SATURDAY_STAY);
    const FareOptionStruct lFareOption1 (lClassPathQ, lFare900, lChangeFee,
                                         isNonRefundable, lSaturdayStay);

    // Add (a copy of) the fare option
    lTS1.addFareOption (lFareOption1);
    //

    // Map of class availabilities: set the availability for the Q
    // booking class (the one corresponding to the fare option) to 8.
    ClassAvailabilityMap_T lClassAvailabilityMap1;
    const Availability_T lAvl1 (8);
    bool hasInsertOfQBeenSuccessful = lClassAvailabilityMap1.
```

```
      insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl1)).second;
    assert (hasInsertOfQBeenSuccessful == true);
    // Add the map to the dedicated list held by the travel solution
    lTS1.addClassAvailabilityMap (lClassAvailabilityMap1);

    // Add the travel solution to the list
    ioTravelSolutionList.push_back (lTS1);

    //
    const std::string lQF12_SegmentDateKey ("QF, 12, 2011-06-10, LHR, SYD, 20:45"
);

    // Add the segment date key to the travel solution
    TravelSolutionStruct lTS2;
    lTS2.addSegment (lQF12_SegmentDateKey);

    // Fare option number 2
    const ClassCode_T lClassPathY (CLASS_CODE_Y);
    const Fare_T lFare1000 (1000);
    const ChangeFees_T lNoChangeFee (NO_CHANGE_FEES);
    const NonRefundable_T isRefundable (NO_NON_REFUNDABLE);
    const FareOptionStruct lFareOption2 (lClassPathY, lFare1000, lNoChangeFee,
                                         isRefundable, lSaturdayStay);

    // Map of class availabilities: set the availability for the Y
    // booking class (the one corresponding to the fare option) to 9.
    ClassAvailabilityMap_T lClassAvailabilityMap2;
    const Availability_T lAvl2 (9);
    const bool hasInsertOfYBeenSuccessful = lClassAvailabilityMap2.
      insert (ClassAvailabilityMap_T::value_type (lClassPathY, lAvl2)).second;
    assert (hasInsertOfYBeenSuccessful == true);
    // Add the map to the dedicated list held by the travel solution
    lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

    // Add (a copy of) the fare option
    lTS2.addFareOption (lFareOption2);

    // Fare option number 3
    const Fare_T lFare920 (920);
    const FareOptionStruct lFareOption3 (lClassPathQ, lFare920, lNoChangeFee,
                                         isNonRefundable, lSaturdayStay);

    // Map of class availabilities: set the availability for the Q
    // booking class (the one corresponding to the fare option) to 9.
    hasInsertOfQBeenSuccessful = lClassAvailabilityMap2.
      insert (ClassAvailabilityMap_T::value_type (lClassPathQ, lAvl2)).second;
    assert (hasInsertOfYBeenSuccessful == true);
    // Add the map to the dedicated list held by the travel solution
    lTS2.addClassAvailabilityMap (lClassAvailabilityMap2);

    // Add (a copy of) the fare option
    lTS2.addFareOption (lFareOption3);

    // Add the travel solution to the list
    ioTravelSolutionList.push_back (lTS2);

  }

  // ////////////////////////////////////////////////////////////////////////
  BookingRequestStruct CmdBomManager::buildSampleBookingRequest() {
    // Origin
    const AirportCode_T lOrigin (AIRPORT_LHR);

    // Destination
    const AirportCode_T lDestination (AIRPORT_SYD);

    // Point of Sale (POS)
```

```
    const CityCode_T lPOS (POS_LHR);

    // Preferred departure date (10-JUN-2011)
    const Date_T lPreferredDepartureDate (2011, boost::gregorian::Jun, 10);

    // Preferred departure time (08:00)
    const Duration_T lPreferredDepartureTime (8, 0, 0);

    // Date of the request (15-MAY-2011)
    const Date_T lRequestDate (2011, boost::gregorian::May, 15);

    // Time of the request (10:00)
    const Duration_T lRequestTime (10, 0, 0);

    // Date-time of the request (made of the date and time above)
    const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

    // Preferred cabin (also named class of service sometimes)
    const CabinCode_T lPreferredCabin (CABIN_ECO);

    // Number of persons in the party
    const PartySize_T lPartySize (3);

    // Channel (direct/indirect, on-line/off-line)
    const ChannelLabel_T lChannel (CHANNEL_DN);

    // Type of the trip (one-way, inbound/outbound of a return trip)
    const TripType_T lTripType (TRIP_TYPE_INBOUND);

    // Duration of the stay (expressed as a number of days)
    const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

    // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
    const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

    // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
    const WTP_T lWTP (DEFAULT_WTP);

    // Value of time, for the customer (expressed in monetary unit per
    // unit of time, e.g., EUR/hour)
    const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

    // Restrictions
    const ChangeFees_T lChangeFees = false;
    const Disutility_T lChangeFeeDisutility = 30;
    const NonRefundable_T lNonRefundable = false;
    const Disutility_T lNonRefundableDisutility = 50;

    // Creation of the booking request structure
    BookingRequestStruct oBookingRequest (lOrigin, lDestination, lPOS,
                                          lPreferredDepartureDate,
                                          lRequestDateTime,
                                          lPreferredCabin,
                                          lPartySize, lChannel,
                                          lTripType, lStayDuration,
                                          lFrequentFlyerType,
                                          lPreferredDepartureTime,
                                          lWTP, lValueOfTime,
                                          lChangeFees, lChangeFeeDisutility,
                                          lNonRefundable,
                                          lNonRefundableDisutility);

    return oBookingRequest;
  }

  // ////////////////////////////////////////////////////////////////////
  BookingRequestStruct CmdBomManager::buildSampleBookingRequestForCRS() {
```

```
    // Origin
    const AirportCode_T lOrigin (AIRPORT_SIN);

    // Destination
    const AirportCode_T lDestination (AIRPORT_BKK);

    // Point of Sale (POS)
    const CityCode_T lPOS (POS_SIN);

    // Preferred departure date (30-JAN-2010)
    const Date_T lPreferredDepartureDate (2010, boost::gregorian::Jan, 30);

    // Preferred departure time (10:00)
    const Duration_T lPreferredDepartureTime (10, 0, 0);

    // Date of the request (22-JAN-2010)
    const Date_T lRequestDate (2010, boost::gregorian::Jan, 22);

    // Time of the request (10:00)
    const Duration_T lRequestTime (10, 0, 0);

    // Date-time of the request (made of the date and time above)
    const DateTime_T lRequestDateTime (lRequestDate, lRequestTime);

    // Preferred cabin (also named class of service sometimes)
    const CabinCode_T lPreferredCabin (CABIN_ECO);

    // Number of persons in the party
    const PartySize_T lPartySize (3);

    // Channel (direct/indirect, on-line/off-line)
    const ChannelLabel_T lChannel (CHANNEL_IN);

    // Type of the trip (one-way, inbound/outbound of a return trip)
    const TripType_T lTripType (TRIP_TYPE_INBOUND);

    // Duration of the stay (expressed as a number of days)
    const DayDuration_T lStayDuration (DEFAULT_STAY_DURATION);

    // Frequent flyer tier (member, silver, gold, platinum, senator, etc)
    const FrequentFlyer_T lFrequentFlyerType (FREQUENT_FLYER_MEMBER);

    // Maximum willing-to-pay (WTP, expressed in monetary unit, e.g., EUR)
    const WTP_T lWTP (DEFAULT_WTP);

    // Value of time, for the customer (expressed in monetary unit per
    // unit of time, e.g., EUR/hour)
    const PriceValue_T lValueOfTime (DEFAULT_VALUE_OF_TIME);

    // Restrictions
    const ChangeFees_T lChangeFees = true;
    const Disutility_T lChangeFeeDisutility = 50;
    const NonRefundable_T lNonRefundable = true;
    const Disutility_T lNonRefundableDisutility = 50;

    // Creation of the booking request structure
    BookingRequestStruct oBookingRequest (lOrigin,
                                          lDestination,
                                          lPOS,
                                          lPreferredDepartureDate,
                                          lRequestDateTime,
                                          lPreferredCabin,
                                          lPartySize, lChannel,
                                          lTripType, lStayDuration,
                                          lFrequentFlyerType,
                                          lPreferredDepartureTime,
                                          lWTP, lValueOfTime,
```

```
                                        lChangeFees, lChangeFeeDisutility,
                                        lNonRefundable,
                                        lNonRefundableDisutility);

    return oBookingRequest;
  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::
  buildPartnershipsSampleInventoryAndRM (BomRoot& ioBomRoot) {

    // Step 0.1: Inventory level
    // Create an Inventory for SQ
    const AirlineCode_T lAirlineCodeSQ ("SQ");
    const InventoryKey lSQKey (lAirlineCodeSQ);
    Inventory& lSQInv = FacBom<Inventory>::instance().create (lSQKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSQInv);
    FacBomManager::linkWithParent (ioBomRoot, lSQInv);

    // Add the airline feature object to the SQ inventory
    const AirlineFeatureKey lAirlineFeatureSQKey (lAirlineCodeSQ);
    AirlineFeature& lAirlineFeatureSQ =
      FacBom<AirlineFeature>::instance().create (lAirlineFeatureSQKey);
    FacBomManager::setAirlineFeature (lSQInv, lAirlineFeatureSQ);
    FacBomManager::linkWithParent (lSQInv, lAirlineFeatureSQ);
    // Link the airline feature object with the top of the BOM tree
    FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureSQ);

    // Create an Inventory for CX
    const AirlineCode_T lAirlineCodeCX ("CX");
    const InventoryKey lCXKey (lAirlineCodeCX);
    Inventory& lCXInv = FacBom<Inventory>::instance().create (lCXKey);
    FacBomManager::addToListAndMap (ioBomRoot, lCXInv);
    FacBomManager::linkWithParent (ioBomRoot, lCXInv);

    // Add the airline feature object to the CX inventory
    const AirlineFeatureKey lAirlineFeatureCXKey (lAirlineCodeCX);
    AirlineFeature& lAirlineFeatureCX =
      FacBom<AirlineFeature>::instance().create (lAirlineFeatureCXKey);
    FacBomManager::setAirlineFeature (lCXInv, lAirlineFeatureCX);
    FacBomManager::linkWithParent (lCXInv, lAirlineFeatureCX);
    // Link the airline feature object with the top of the BOM tree
    FacBomManager::addToListAndMap (ioBomRoot, lAirlineFeatureCX);

    // ////// SQ ///////
    // Step 0.2: Flight-date level
    // Create a FlightDate (SQ11/08-MAR-2010) for SQ's Inventory
    FlightNumber_T lFlightNumber = 11;
    Date_T lDate (2010, 3, 8);
    FlightDateKey lFlightDateKey (lFlightNumber, lDate);

    FlightDate& lSQ11_20100308_FD =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lSQInv, lSQ11_20100308_FD);
    FacBomManager::linkWithParent (lSQInv, lSQ11_20100308_FD);

    // Create a (mkt) FlightDate (SQ1200/08-MAR-2010) for SQ's Inventory
    FlightNumber_T lMktFlightNumber = 1200;
    //lDate = Date_T (2010, 3, 8);
    FlightDateKey lMktFlightDateKey (lMktFlightNumber, lDate);

    FlightDate& lSQ1200_20100308_FD =
      FacBom<FlightDate>::instance().create (lMktFlightDateKey);
    FacBomManager::addToListAndMap (lSQInv, lSQ1200_20100308_FD);
    FacBomManager::linkWithParent (lSQInv, lSQ1200_20100308_FD);

    // Display the flight-date
```

```
    // STDAIR_LOG_DEBUG ("FlightDate: " << lBA9_20110610_FD.toString());

    // Step 0.3: Segment-date level
    // Create a first SegmentDate (SIN-BKK) for SQ's Inventory
    const AirportCode_T lSIN ("SIN");
    const AirportCode_T lBKK ("BKK");
    const DateOffset_T l1Day (1);
    const DateOffset_T l2Days (2);
    const Duration_T l0820 (8, 20, 0);
    const Duration_T l1100 (11, 0, 0);
    const Duration_T l0340 (3, 40, 0);
    SegmentDateKey lSegmentDateKey (lSIN, lBKK);

    SegmentDate& lSINBKKSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINBKKSegment);
    FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINBKKSegment);

    // Add the routing leg key to the SIN-BKK segment.
    const std::string lSQSINRoutingLegStr = "SQ;11;2010-Mar-8;SIN";
    lSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

    // Fill the SegmentDate content
    lSINBKKSegment.setBoardingDate (lDate);
    lSINBKKSegment.setOffDate (lDate);
    lSINBKKSegment.setBoardingTime (l0820);
    lSINBKKSegment.setOffTime (l1100);
    lSINBKKSegment.setElapsedTime (l0340);

    // Create a second (mkt) SegmentDate (BKK-HKG) for SQ's Inventory
    const AirportCode_T lHKG ("HKG");
    const Duration_T l1200 (12, 0, 0);
    const Duration_T l1540 (15, 40, 0);
    const Duration_T l0240 (2, 40, 0);
    SegmentDateKey lMktSegmentDateKey (lBKK, lHKG);

    SegmentDate& lMktBKKHKGSegment =
      FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
    FacBomManager::addToListAndMap (lSQ1200_20100308_FD, lMktBKKHKGSegment);
    FacBomManager::linkWithParent (lSQ1200_20100308_FD, lMktBKKHKGSegment);

    // Add the routing leg key CX;12;2010-Mar-8;BKK to the marketing
    // SQ;1200;2010-Mar-8;BKK-HKG segment.
    const std::string lCXBKKRoutingLegStr = "CX;12;2010-Mar-8;BKK";
    lMktBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

    // Fill the (mkt) SegmentDate content
    lMktBKKHKGSegment.setBoardingDate (lDate);
    lMktBKKHKGSegment.setOffDate (lDate);
    lMktBKKHKGSegment.setBoardingTime (l1200);
    lMktBKKHKGSegment.setOffTime (l1540);
    lMktBKKHKGSegment.setElapsedTime (l0240);

    // Step 0.4: Leg-date level
    // Create a first LegDate (SIN) for SQ's Inventory
    LegDateKey lLegDateKey (lSIN);

    LegDate& lSINLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lSQ11_20100308_FD, lSINLeg);
    FacBomManager::linkWithParent (lSQ11_20100308_FD, lSINLeg);

    // Fill the LegDate content
    lSINLeg.setOffPoint (lBKK);
    lSINLeg.setBoardingDate (lDate);
    lSINLeg.setOffDate (lDate);
    lSINLeg.setBoardingTime (l0820);
    lSINLeg.setOffTime (l1100);
```

```
    lSINLeg.setElapsedTime (l0340);

    // Step 0.5: segment-cabin level
    // Create a SegmentCabin (Y) for the Segment SIN-BKK of SQ's Inventory
    const CabinCode_T lY ("Y");
    SegmentCabinKey lYSegmentCabinKey (lY);

    SegmentCabin& lSINBKKSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lSINBKKSegment, lSINBKKSegmentYCabin);
    FacBomManager::linkWithParent (lSINBKKSegment, lSINBKKSegmentYCabin);

    // Create a SegmentCabin (Y) for the (mkt) Segment BKK-HKG of SQ's Inventory
    SegmentCabin& lMktBKKHKGSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);
    FacBomManager::linkWithParent (lMktBKKHKGSegment, lMktBKKHKGSegmentYCabin);


    // Step 0.6: leg-cabin level
    // Create a LegCabin (Y) for the Leg SIN-BKK on SQ's Inventory
    LegCabinKey lYLegCabinKey (lY);

    LegCabin& lSINLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lSINLeg, lSINLegYCabin);
    FacBomManager::linkWithParent (lSINLeg, lSINLegYCabin);

    CabinCapacity_T lCapacity (100);
    lSINLegYCabin.setCapacities (lCapacity);
    lSINLegYCabin.setAvailabilityPool (lCapacity);


    // Step 0.7: fare family level
    // Create a FareFamily (1) for the Segment SIN-BKK, cabin Y on SQ's Inv
    const FamilyCode_T l1 ("EcoSaver");
    FareFamilyKey l1FareFamilyKey (l1);

    FareFamily& lSINBKKSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1Family);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin,
                                   lSINBKKSegmentYCabin1Family);

    // Create a FareFamily (1) for the (mkt) Segment BKK-HKG, cabin Y on SQ's Inv

    FareFamily& lMktBKKHKGSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                    lMktBKKHKGSegmentYCabin1Family);
    FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin,
                                    lMktBKKHKGSegmentYCabin1Family);

    // Step 0.8: booking class level
    // Create a BookingClass (Y) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    BookingClassKey lYBookingClassKey (lY);

    BookingClass& lSINBKKSegmentYCabin1FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyYClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyYClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
```

```
                                    lSINBKKSegmentYCabin1FamilyYClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin1FamilyYClass);

    lSINBKKSegmentYCabin1FamilyYClass.setYield(700);

    // Create a BookingClass (Y) for the (mkt) Segment BKK-HKG, cabin Y,
    // fare family 1 on SQ's Inv
    BookingClass& lMktBKKHKGSegmentYCabin1FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                    lMktBKKHKGSegmentYCabin1FamilyYClass);
    FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin1Family,
                                   lMktBKKHKGSegmentYCabin1FamilyYClass);

    FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                    lMktBKKHKGSegmentYCabin1FamilyYClass);
    FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                    lMktBKKHKGSegmentYCabin1FamilyYClass);

    lMktBKKHKGSegmentYCabin1FamilyYClass.setYield(700);


    // Create a BookingClass (M) for the Segment SIN-BKK, cabin Y,
    // fare family 1 on SQ's Inv
    const ClassCode_T lM ("M");
    BookingClassKey lMBookingClassKey (lM);

    BookingClass& lSINBKKSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin1Family,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lSINBKKSegmentYCabin1Family,
                                   lSINBKKSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lSINBKKSegmentYCabin,
                                    lSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lSINBKKSegment,
                                    lSINBKKSegmentYCabin1FamilyMClass);

    lSINBKKSegmentYCabin1FamilyMClass.setYield(500);

    // Create a BookingClass (M) for the (mkt) Segment BKK-HKG, cabin Y,
    // fare family 1 on SQ's Inv
    BookingClass& lMktBKKHKGSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin1Family,
                                    lMktBKKHKGSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lMktBKKHKGSegmentYCabin1Family,
                                   lMktBKKHKGSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lMktBKKHKGSegmentYCabin,
                                    lMktBKKHKGSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lMktBKKHKGSegment,
                                    lMktBKKHKGSegmentYCabin1FamilyMClass);

    lMktBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

    /* ========================================================================
    ========= */

    // Step 1.0: O&D level
    // Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK-SQ1200/08-MAR-2010/BKK-HKG)
    // for SQ's Inventory
    OnDString_T lSQSINBKKOnDStr = "SQ;11,2010-Mar-08;SIN,BKK";
    OnDString_T lMktSQBKKHKGOnDStr = "SQ;1200,2010-Mar-08;BKK,HKG";
    OnDStringList_T lOnDStringList;
```

```
    lOnDStringList.push_back (lSQSINBKKOnDStr);
    lOnDStringList.push_back (lMktSQBKKHKGOnDStr);

    OnDDateKey lOnDDateKey (lOnDStringList);
    OnDDate& lSQ_SINHKG_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lSQInv, lSQ_SINHKG_OnDDate);
    FacBomManager::linkWithParent (lSQInv, lSQ_SINHKG_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lSINBKKSegment);
    FacBomManager::addToListAndMap (lSQ_SINHKG_OnDDate, lMktBKKHKGSegment);

    // Add total forecast info for cabin Y.
    const MeanStdDevPair_T lMean60StdDev6 (60.0, 6.0);
    const WTP_T lWTP750 = 750.0;
    const WTPDemandPair_T lWTP750Mean60StdDev6 (lWTP750, lMean60StdDev6);
    lSQ_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

    // Add demand info (optional).
    // 2 legs here, so 2 CabinClassPair to add in the list.
    // Fist leg: cabin Y, class M.
    CabinClassPair_T lCC_YM1 (lY,lM);
    // Second leg: cabin Y, class M too.
    CabinClassPair_T lCC_YM2 (lY,lM);
    CabinClassPairList_T lCabinClassPairList;
    lCabinClassPairList.push_back(lCC_YM1);
    lCabinClassPairList.push_back(lCC_YM2);
    const MeanStdDevPair_T lMean20StdDev2 (20.0, 2.0);
    const Yield_T lYield850 = 850.0;
    const YieldDemandPair_T lYield850Mean20StdDev2 (lYield850, lMean20StdDev2);
    lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList, lYield850Mean20
StdDev2);

    CabinClassPair_T lCC_YY1 (lY,lY);
    CabinClassPair_T lCC_YY2 (lY,lY);
    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YY1);
    lCabinClassPairList.push_back(lCC_YY2);
    const MeanStdDevPair_T lMean10StdDev1 (10.0, 1.0);
    const Yield_T lYield1200 = 1200.0;
    const YieldDemandPair_T lYield1200Mean10StdDev1 (lYield1200,
                                                     lMean10StdDev1);
    lSQ_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield1200Mean10StdDev1);

    // Create an O&D Date (SQ11/08-MAR-2010/SIN-BKK) for SQ's Inventory
    lOnDStringList.clear();
    lOnDStringList.push_back (lSQSINBKKOnDStr);

    lOnDDateKey = OnDDateKey(lOnDStringList);
    OnDDate& lSQ_SINBKK_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lSQInv, lSQ_SINBKK_OnDDate);
    FacBomManager::linkWithParent (lSQInv, lSQ_SINBKK_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lSQ_SINBKK_OnDDate, lSINBKKSegment);

    // Add total forecast info for cabin Y.
    const WTP_T lWTP400 = 400.0;
    const WTPDemandPair_T lWTP400Mean60StdDev6 (lWTP400, lMean60StdDev6);
    lSQ_SINBKK_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

    // Add demand info (optional).
```

```
    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YM1);
    const MeanStdDevPair_T lMean20StdDev1 (20.0, 1.0);
    const Yield_T lYield500 = 500.0;
    const YieldDemandPair_T lYield500Mean20StdDev1 (lYield500, lMean20StdDev1);
    lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield500Mean20StdDev1);

    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YY1);
    const Yield_T lYield700 = 700.0;
    const YieldDemandPair_T lYield700Mean20StdDev1 (lYield700, lMean10StdDev1 );
    lSQ_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield700Mean20StdDev1);

    /*****************************************************************************
    // Create an O&D Date (SQ1200/08-MAR-2010/BKK-HKG) for SQ's Inventory
    lFullKeyList.clear();
    lFullKeyList.push_back (lMktSQBKKHKGFullKeyStr);

    lOnDDateKey = OnDDateKey(lFullKeyList);
    OnDDate& lMktSQ_BKKHKG_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lSQInv, lMktSQ_BKKHKG_OnDDate);
    FacBomManager::linkWithParent (lSQInv, lMktSQ_BKKHKG_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lMktSQ_BKKHKG_OnDDate, lMktBKKHKGSegment);

    // Demand info is not added for purely marketed O&Ds
    // Add demand info
    // lCabinClassPairList.clear();
    // lCabinClassPairList.push_back(lCC_YM2);
    // lMktSQ_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
    ******************************************************************************/


    // ////// CX ///////
    // Step 0.2: Flight-date level
    // Create a FlightDate (CX12/08-MAR-2010) for CX's Inventory
    lFlightNumber = 12;
    //lDate = Date_T (2010, 2, 8);
    lFlightDateKey = FlightDateKey (lFlightNumber, lDate);

    FlightDate& lCX12_20100308_FD =
      FacBom<FlightDate>::instance().create (lFlightDateKey);
    FacBomManager::addToListAndMap (lCXInv, lCX12_20100308_FD);
    FacBomManager::linkWithParent (lCXInv, lCX12_20100308_FD);

    // Create a (mkt) FlightDate (CX1100/08-FEB-2010) for CX's Inventory
    lFlightNumber = 1100;
    //lDate = Date_T (2010, 2, 8);
    lMktFlightDateKey = FlightDateKey (lFlightNumber, lDate);

    FlightDate& lCX1100_20100308_FD =
      FacBom<FlightDate>::instance().create (lMktFlightDateKey);
    FacBomManager::addToListAndMap (lCXInv, lCX1100_20100308_FD);
    FacBomManager::linkWithParent (lCXInv, lCX1100_20100308_FD);

    // Display the flight-date
    // STDAIR_LOG_DEBUG ("FlightDate: " << lAF084_20110320_FD.toString());

    // Step 0.3: Segment-date level
```

```
    // Create a SegmentDate BKK-HKG for CX's Inventory

    lSegmentDateKey = SegmentDateKey (lBKK, lHKG);

    SegmentDate& lBKKHKGSegment =
      FacBom<SegmentDate>::instance().create (lSegmentDateKey);
    FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKHKGSegment);
    FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKHKGSegment);

    // Add the routing leg key to the marketing BKK-HKG segment.
    lBKKHKGSegment.addLegKey (lCXBKKRoutingLegStr);

    // Fill the SegmentDate content
    lBKKHKGSegment.setBoardingDate (lDate);
    lBKKHKGSegment.setOffDate (lDate);
    lBKKHKGSegment.setBoardingTime (l1200);
    lBKKHKGSegment.setOffTime (l1540);
    lBKKHKGSegment.setElapsedTime (l0240);

    // Create a second (mkt) SegmentDate (SIN-BKK) for CX's Inventory
    lMktSegmentDateKey = SegmentDateKey (lSIN, lBKK);

    SegmentDate& lMktSINBKKSegment =
      FacBom<SegmentDate>::instance().create (lMktSegmentDateKey);
    FacBomManager::addToListAndMap (lCX1100_20100308_FD, lMktSINBKKSegment);
    FacBomManager::linkWithParent (lCX1100_20100308_FD, lMktSINBKKSegment);

    // Add the routing leg key SQ;11;2010-Mar-8;SIN to the marketing
    // CX;1100;2010-Mar-8;SIN-BKK segment.
    lMktSINBKKSegment.addLegKey (lSQSINRoutingLegStr);

    // Fill the (mkt) SegmentDate content
    lMktSINBKKSegment.setBoardingDate (lDate);
    lMktSINBKKSegment.setOffDate (lDate);
    lMktSINBKKSegment.setBoardingTime (l0820);
    lMktSINBKKSegment.setOffTime (l1100);
    lMktSINBKKSegment.setElapsedTime (l0340);

    // Step 0.4: Leg-date level
    // Create a LegDate (BKK) for CX's Inventory
    lLegDateKey = LegDateKey (lBKK);

    LegDate& lBKKLeg = FacBom<LegDate>::instance().create (lLegDateKey);
    FacBomManager::addToListAndMap (lCX12_20100308_FD, lBKKLeg);
    FacBomManager::linkWithParent (lCX12_20100308_FD, lBKKLeg);

    // Fill the LegDate content
    lBKKLeg.setOffPoint (lHKG);
    lBKKLeg.setBoardingDate (lDate);
    lBKKLeg.setOffDate (lDate);
    lBKKLeg.setBoardingTime (l1200);
    lBKKLeg.setOffTime (l1540);
    lBKKLeg.setElapsedTime (l0240);

    // Display the leg-date
    // STDAIR_LOG_DEBUG ("LegDate: " << lCDGLeg.toString());

    // Step 0.5: segment-cabin level
    // Create a SegmentCabin (Y) for the Segment BKK-HKG of CX's Inventory
    SegmentCabin& lBKKHKGSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
    FacBomManager::addToListAndMap (lBKKHKGSegment, lBKKHKGSegmentYCabin);
    FacBomManager::linkWithParent (lBKKHKGSegment, lBKKHKGSegmentYCabin);

    // Create a SegmentCabin (Y) for the (mkt) Segment SIN-BKK of CX's Inventory
    SegmentCabin& lMktSINBKKSegmentYCabin =
      FacBom<SegmentCabin>::instance().create (lYSegmentCabinKey);
```

```
    FacBomManager::addToListAndMap (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);
    FacBomManager::linkWithParent (lMktSINBKKSegment, lMktSINBKKSegmentYCabin);

    // Step 0.6: leg-cabin level
    // Create a LegCabin (Y) for the Leg BKK-HKG on CX's Inventory
    LegCabin& lBKKLegYCabin =
      FacBom<LegCabin>::instance().create (lYLegCabinKey);
    FacBomManager::addToListAndMap (lBKKLeg, lBKKLegYCabin);
    FacBomManager::linkWithParent (lBKKLeg, lBKKLegYCabin);

    lCapacity = CabinCapacity_T(100);
    lBKKLegYCabin.setCapacities (lCapacity);
    lBKKLegYCabin.setAvailabilityPool (lCapacity);

    // Step 0.7: fare family level
    // Create a fareFamily (1) for the Segment BKK-HKG, cabin Y on CX's Inv
    FareFamily& lBKKHKGSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                    lBKKHKGSegmentYCabin1Family);
    FacBomManager::linkWithParent (lBKKHKGSegmentYCabin,
                                   lBKKHKGSegmentYCabin1Family);

    // Create a FareFamily (1) for the (mkt) Segment SIN-BKK, cabin Y on CX's Inv

    FareFamily& lMktSINBKKSegmentYCabin1Family =
      FacBom<FareFamily>::instance().create (l1FareFamilyKey);
    FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                    lMktSINBKKSegmentYCabin1Family);
    FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin,
                                   lMktSINBKKSegmentYCabin1Family);


    // Step 0.8: booking class level
    // Create a BookingClass (Y) for the
    // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
    BookingClass& lBKKHKGSegmentYCabin1FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                    lBKKHKGSegmentYCabin1FamilyYClass);
    FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                                   lBKKHKGSegmentYCabin1FamilyYClass);

    FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                    lBKKHKGSegmentYCabin1FamilyYClass);
    FacBomManager::addToListAndMap (lBKKHKGSegment,
                                    lBKKHKGSegmentYCabin1FamilyYClass);

    lBKKHKGSegmentYCabin1FamilyYClass.setYield(700);

    // Create a BookingClass (Y) for the (mkt) Segment SIN-BKK, cabin Y,
    // fare family 1 on CX's Inv
    BookingClass& lMktSINBKKSegmentYCabin1FamilyYClass =
      FacBom<BookingClass>::instance().create (lYBookingClassKey);
    FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                    lMktSINBKKSegmentYCabin1FamilyYClass);
    FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                                   lMktSINBKKSegmentYCabin1FamilyYClass);

    FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                    lMktSINBKKSegmentYCabin1FamilyYClass);
    FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                    lMktSINBKKSegmentYCabin1FamilyYClass);

    lMktSINBKKSegmentYCabin1FamilyYClass.setYield(700);

    //Create a BookingClass (M) for the
```

```
    // Segment BKK-HKG, cabin Y, fare family 1 on CX's Inv
    BookingClass& lBKKHKGSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin1Family,
                                    lBKKHKGSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lBKKHKGSegmentYCabin1Family,
                                   lBKKHKGSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lBKKHKGSegmentYCabin,
                                    lBKKHKGSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lBKKHKGSegment,
                                    lBKKHKGSegmentYCabin1FamilyMClass);

    lBKKHKGSegmentYCabin1FamilyMClass.setYield(500);

    // Create a BookingClass (M) for the (mkt) Segment SIN-BKK, cabin Y,
    // fare family 1 on CX's Inv
    BookingClass& lMktSINBKKSegmentYCabin1FamilyMClass =
      FacBom<BookingClass>::instance().create (lMBookingClassKey);
    FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin1Family,
                                    lMktSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::linkWithParent (lMktSINBKKSegmentYCabin1Family,
                                   lMktSINBKKSegmentYCabin1FamilyMClass);

    FacBomManager::addToListAndMap (lMktSINBKKSegmentYCabin,
                                    lMktSINBKKSegmentYCabin1FamilyMClass);
    FacBomManager::addToListAndMap (lMktSINBKKSegment,
                                    lMktSINBKKSegmentYCabin1FamilyMClass);

    lMktSINBKKSegmentYCabin1FamilyMClass.setYield(500);

    /* ========================================================================
    ========= */

    // Step 1.0: O&D level
    // Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK-CX12/08-MAR-2010/BKK-HKG) f
    or CX's Inventory
    OnDString_T lMktCXSINBKKOnDStr = "CX;1100,2010-Mar-08;SIN,BKK";
    OnDString_T lCXBKKHKGOnDStr = "CX;12,2010-Mar-08;BKK,HKG";
    lOnDStringList.clear();
    lOnDStringList.push_back (lMktCXSINBKKOnDStr);
    lOnDStringList.push_back (lCXBKKHKGOnDStr);

    lOnDDateKey = OnDDateKey(lOnDStringList);
    OnDDate& lCX_SINHKG_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lCXInv, lCX_SINHKG_OnDDate);
    FacBomManager::linkWithParent (lCXInv, lCX_SINHKG_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lMktSINBKKSegment);
    FacBomManager::addToListAndMap (lCX_SINHKG_OnDDate, lBKKHKGSegment);

    // Add total forecast info for cabin Y.
    lCX_SINHKG_OnDDate.setTotalForecast (lY, lWTP750Mean60StdDev6);

    // Add demand info
    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YM1);
    lCabinClassPairList.push_back(lCC_YM2);
    lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield850Mean20StdDev2);

    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YY1);
    lCabinClassPairList.push_back(lCC_YY2);
```

```
    lCX_SINHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield1200Mean10StdDev1);

    /*****************************************************************************
    // Create an O&D Date (CX1100/08-MAR-2010/SIN-BKK) for CX's Inventory
    lFullKeyList.clear();
    lFullKeyList.push_back (lMktCXSINBKKFullKeyStr);

    lOnDDateKey = OnDDateKey(lFullKeyList);
    OnDDate& lMktCX_SINBKK_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lCXInv, lMktCX_SINBKK_OnDDate);
    FacBomManager::linkWithParent (lCXInv, lMktCX_SINBKK_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lMktCX_SINBKK_OnDDate, lMktSINBKKSegment);

    // Demand info is not added for purely marketed O&Ds
    // Add demand info
    // lCabinClassPairList.clear();
    // lCabinClassPairList.push_back(lCC_YM1);
    // lMktCX_SINBKK_OnDDate.setDemandInformation (lCabinClassPairList, 500.0, 20.0, 1.0);
    *****************************************************************************/

    // Create an O&D Date (CX12/08-FEB-2010/BKK-HKG) for CX's Inventory
    lOnDStringList.clear();
    lOnDStringList.push_back (lCXBKKHKGOnDStr);

    lOnDDateKey = OnDDateKey(lOnDStringList);
    OnDDate& lCX_BKKHKG_OnDDate =
      FacBom<OnDDate>::instance().create (lOnDDateKey);
    // Link to the inventory
    FacBomManager::addToListAndMap (lCXInv, lCX_BKKHKG_OnDDate);
    FacBomManager::linkWithParent (lCXInv, lCX_BKKHKG_OnDDate);

    // Add the segments
    FacBomManager::addToListAndMap (lCX_BKKHKG_OnDDate, lBKKHKGSegment);

    // Add total forecast info for cabin Y.
    lCX_BKKHKG_OnDDate.setTotalForecast (lY, lWTP400Mean60StdDev6);

    // Add demand info
    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YM2);
    lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield500Mean20StdDev1);

    lCabinClassPairList.clear();
    lCabinClassPairList.push_back(lCC_YY2);
    const YieldDemandPair_T lYield700Mean10StdDev1 (lYield700, lMean10StdDev1 );
    lCX_BKKHKG_OnDDate.setDemandInformation (lCabinClassPairList,
                                             lYield700Mean10StdDev1);

    /*=============================================================================
      ==============================================================================
      ==============================================================================*/
    // Schedule:
    // SQ:
    // Step 1: flight period level
    // Create a flight period for SQ11:
```

```
    const DoWStruct lDoWSrtuct ("1111111");
    const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 8);
    const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 9);
    const DatePeriod_T lDatePeriod (lDateRangeStart, lDateRangeEnd);
    const PeriodStruct lPeriodStruct (lDatePeriod,lDoWSrtuct);

    lFlightNumber = FlightNumber_T (11);

    FlightPeriodKey lFlightPeriodKey (lFlightNumber, lPeriodStruct);

    FlightPeriod& lSQ11FlightPeriod =
      FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
    FacBomManager::addToListAndMap (lSQInv, lSQ11FlightPeriod);
    FacBomManager::linkWithParent (lSQInv, lSQ11FlightPeriod);

    // Step 2: segment period level
    // Create a segment period for SIN-BKK:

    SegmentPeriodKey lSegmentPeriodKey (lSIN, lBKK);

    SegmentPeriod& lSINBKKSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
    FacBomManager::addToListAndMap (lSQ11FlightPeriod, lSINBKKSegmentPeriod);
    FacBomManager::linkWithParent (lSQ11FlightPeriod, lSINBKKSegmentPeriod);

    lSINBKKSegmentPeriod.setBoardingTime (l0820);
    lSINBKKSegmentPeriod.setOffTime (l1100);
    lSINBKKSegmentPeriod.setElapsedTime (l0340);
    ClassList_String_T lYM ("YM");
    lSINBKKSegmentPeriod.addCabinBookingClassList (lY,lYM);

    // CX:
    // Step 1: flight period level
    // Create a flight period for CX12:
    lFlightNumber = FlightNumber_T (12);

    lFlightPeriodKey = FlightPeriodKey(lFlightNumber, lPeriodStruct);

    FlightPeriod& lCX12FlightPeriod =
      FacBom<FlightPeriod>::instance().create (lFlightPeriodKey);
    FacBomManager::addToListAndMap (lCXInv, lCX12FlightPeriod);
    FacBomManager::linkWithParent (lCXInv, lCX12FlightPeriod);

    // Step 2: segment period level
    // Create a segment period for BKK-HKG:

    lSegmentPeriodKey = SegmentPeriodKey (lBKK, lHKG);

    SegmentPeriod& lBKKHKGSegmentPeriod =
      FacBom<SegmentPeriod>::instance().create (lSegmentPeriodKey);
    FacBomManager::addToListAndMap (lCX12FlightPeriod, lBKKHKGSegmentPeriod);
    FacBomManager::linkWithParent (lCX12FlightPeriod, lBKKHKGSegmentPeriod);

    lBKKHKGSegmentPeriod.setBoardingTime (l1200);
    lBKKHKGSegmentPeriod.setOffTime (l1540);
    lBKKHKGSegmentPeriod.setElapsedTime (l0240);
    lBKKHKGSegmentPeriod.addCabinBookingClassList (lY,lYM);

  }

  // ////////////////////////////////////////////////////////////////////
  void CmdBomManager::buildPartnershipsSamplePricing (BomRoot& ioBomRoot) {


    /*=========================================================================
    ========*/
    // First airport pair SIN-BKK.
```

```
    // Set the airport-pair primary key.
    AirportPairKey lAirportPairKey ("SIN", "BKK");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lSINBKKAirportPair =
      FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSINBKKAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lSINBKKAirportPair);

    // Set the fare date-period primary key.
    const Date_T lDateRangeStart (2010, boost::gregorian::Mar, 01);
    const Date_T lDateRangeEnd (2010, boost::gregorian::Mar, 31);
    const DatePeriod_T lDateRange (lDateRangeStart, lDateRangeEnd);
    const DatePeriodKey lDatePeriodKey (lDateRange);

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lSINBKKDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lSINBKKAirportPair, lSINBKKDatePeriod);
    FacBomManager::linkWithParent (lSINBKKAirportPair, lSINBKKDatePeriod);

    // Set the point-of-sale-channel primary key.
    PosChannelKey lPosChannelKey ("SIN","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lSINPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINBKKDatePeriod, lSINPosChannel);
    FacBomManager::linkWithParent (lSINBKKDatePeriod, lSINPosChannel);

    // Set the fare time-period primary key.
    const Time_T lTimeRangeStart (0, 0, 0);
    const Time_T lTimeRangeEnd (23, 0, 0);
    const TimePeriodKey lFareTimePeriodKey (lTimeRangeStart,
                                            lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINBKKFareTimePeriod =
      FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
    FacBomManager::addToListAndMap (lSINPosChannel, lSINBKKFareTimePeriod);
    FacBomManager::linkWithParent (lSINPosChannel, lSINBKKFareTimePeriod);

    // Generate the FareRule
    const FareFeaturesKey lFareFeaturesKey (TRIP_TYPE_ONE_WAY,
                                            NO_ADVANCE_PURCHASE,
                                            SATURDAY_STAY,
                                            CHANGE_FEES,
                                            NON_REFUNDABLE,
                                            NO_STAY_DURATION);

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lSINBKKFareFeatures =
      FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
    FacBomManager::addToListAndMap (lSINBKKFareTimePeriod, lSINBKKFareFeatures);
    FacBomManager::linkWithParent (lSINBKKFareTimePeriod, lSINBKKFareFeatures);

    // Generate Segment Features and link them to their FareRule.
    AirlineCodeList_T lSQAirlineCodeList;
    lSQAirlineCodeList.push_back ("SQ");

    ClassList_StringList_T lYClassCodeList;
    lYClassCodeList.push_back ("Y");
    const AirlineClassListKey lSQAirlineYClassListKey (lSQAirlineCodeList,
                                                       lYClassCodeList);

    ClassList_StringList_T lMClassCodeList;
    lMClassCodeList.push_back ("M");
```

```
    const AirlineClassListKey lSQAirlineMClassListKey (lSQAirlineCodeList,
                                                       lMClassCodeList);

    // Create the AirlineClassListKey and link it to the FareFeatures object.
    AirlineClassList& lSQAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
    lSQAirlineYClassList.setFare(700);
    FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineYClassList);
    FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineYClassList);

    AirlineClassList& lSQAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
    lSQAirlineMClassList.setFare(500);
    FacBomManager::addToListAndMap (lSINBKKFareFeatures, lSQAirlineMClassList);
    FacBomManager::linkWithParent (lSINBKKFareFeatures, lSQAirlineMClassList);

    /*==========================================================================
    ========*/
    // Second airport pair BKK-HKG.
    // Set the airport-pair primary key.
    lAirportPairKey = AirportPairKey ("BKK", "HKG");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lBKKHKGAirportPair =
      FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lBKKHKGAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lBKKHKGAirportPair);

    // Set the fare date-period primary key.
    // Use the same as previously.

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lBKKHKGDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lBKKHKGAirportPair, lBKKHKGDatePeriod);
    FacBomManager::linkWithParent (lBKKHKGAirportPair, lBKKHKGDatePeriod);

    // Set the point-of-sale-channel primary key.
    lPosChannelKey  = PosChannelKey("BKK","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lBKKPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lBKKPosChannel);
    FacBomManager::linkWithParent (lBKKHKGDatePeriod, lBKKPosChannel);

    // Set the fare time-period primary key.
    // Use the same as previously.

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lBKKHKGFareTimePeriod =
      FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
    FacBomManager::addToListAndMap (lBKKPosChannel, lBKKHKGFareTimePeriod);
    FacBomManager::linkWithParent (lBKKPosChannel, lBKKHKGFareTimePeriod);

    // Generate the FareRule
    // Use the same key as previously.

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lBKKHKGFareFeatures =
      FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
    FacBomManager::addToListAndMap (lBKKHKGFareTimePeriod, lBKKHKGFareFeatures);
    FacBomManager::linkWithParent (lBKKHKGFareTimePeriod, lBKKHKGFareFeatures);

    // Generate Segment Features and link them to their FareRule.
    AirlineCodeList_T lCXAirlineCodeList;
    lCXAirlineCodeList.push_back ("CX");
```

```
    const AirlineClassListKey lCXAirlineYClassListKey (lCXAirlineCodeList,
                                                       lYClassCodeList);

    const AirlineClassListKey lCXAirlineMClassListKey (lCXAirlineCodeList,
                                                       lMClassCodeList);

    // Create the AirlineClassListKey and link it to the FareFeatures object.
    AirlineClassList& lCXAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
    lCXAirlineYClassList.setFare(700);
    FacBomManager::addToListAndMap (lBKKHKGFareFeatures, lCXAirlineYClassList);
    FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineYClassList);

    AirlineClassList& lCXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
    lCXAirlineMClassList.setFare(500);
    FacBomManager::addToListAndMap (lBKKHKGFareFeatures, lCXAirlineMClassList);
    FacBomManager::linkWithParent (lBKKHKGFareFeatures, lCXAirlineMClassList);

    /*==========================================================================
    ========*/
    // Third airport pair SIN-HKG.
    // Set the airport-pair primary key.
    lAirportPairKey = AirportPairKey ("SIN", "HKG");

    // Create the AirportPairKey object and link it to the ioBomRoot object.
    AirportPair& lSINHKGAirportPair =
      FacBom<AirportPair>::instance().create (lAirportPairKey);
    FacBomManager::addToListAndMap (ioBomRoot, lSINHKGAirportPair);
    FacBomManager::linkWithParent (ioBomRoot, lSINHKGAirportPair);

    // Set the fare date-period primary key.
    // Use the same as previously.

    // Create the DatePeriodKey object and link it to the PosChannel object.
    DatePeriod& lSINHKGDatePeriod =
      FacBom<DatePeriod>::instance().create (lDatePeriodKey);
    FacBomManager::addToListAndMap (lSINHKGAirportPair, lSINHKGDatePeriod);
    FacBomManager::linkWithParent (lSINHKGAirportPair, lSINHKGDatePeriod);

    // Set the point-of-sale-channel primary key.
    lPosChannelKey = PosChannelKey("SIN","IN");

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lOnDSINPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINHKGDatePeriod, lOnDSINPosChannel);
    FacBomManager::linkWithParent (lSINHKGDatePeriod, lOnDSINPosChannel);

    // Set the fare time-period primary key.
    // Use the same as previously.

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINHKGFareTimePeriod =
      FacBom<TimePeriod>::instance().create (lFareTimePeriodKey);
    FacBomManager::addToListAndMap (lOnDSINPosChannel, lSINHKGFareTimePeriod);
    FacBomManager::linkWithParent (lOnDSINPosChannel, lSINHKGFareTimePeriod);

    // Generate the FareRule
    // Use the same key as previously.

    // Create the FareFeaturesKey and link it to the TimePeriod object.
    FareFeatures& lSINHKGFareFeatures =
      FacBom<FareFeatures>::instance().create (lFareFeaturesKey);
    FacBomManager::addToListAndMap (lSINHKGFareTimePeriod, lSINHKGFareFeatures);
    FacBomManager::linkWithParent (lSINHKGFareTimePeriod, lSINHKGFareFeatures);
```

```

    // Generate Segment Features and link them to their FareRule.
    AirlineCodeList_T lSQ_CXAirlineCodeList;
    lSQ_CXAirlineCodeList.push_back ("SQ");
    lSQ_CXAirlineCodeList.push_back ("CX");

    ClassList_StringList_T lY_YClassCodeList;
    lY_YClassCodeList.push_back ("Y");
    lY_YClassCodeList.push_back ("Y");
    const AirlineClassListKey lSQ_CXAirlineYClassListKey (lSQ_CXAirlineCodeList,
                                                          lY_YClassCodeList);

    ClassList_StringList_T lM_MClassCodeList;
    lM_MClassCodeList.push_back ("M");
    lM_MClassCodeList.push_back ("M");
    const AirlineClassListKey lSQ_CXAirlineMClassListKey (lSQ_CXAirlineCodeList,
                                                          lM_MClassCodeList);

    // Create the AirlineClassListKey and link it to the FareFeatures object.
    AirlineClassList& lSQ_CXAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
    lSQ_CXAirlineYClassList.setFare(1200);
    FacBomManager::addToListAndMap (lSINHKGFareFeatures,
                                    lSQ_CXAirlineYClassList);
    FacBomManager::linkWithParent (lSINHKGFareFeatures,
                                   lSQ_CXAirlineYClassList);

    AirlineClassList& lSQ_CXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
    lSQ_CXAirlineMClassList.setFare(850);
    FacBomManager::addToListAndMap (lSINHKGFareFeatures,
                                    lSQ_CXAirlineMClassList);
    FacBomManager::linkWithParent (lSINHKGFareFeatures,
                                   lSQ_CXAirlineMClassList);



    /*==========================================================================
========*/

    // Use the same airport pair, and date period for adding SQ SIN-BKK yields.

    // Set the point-of-sale-channel primary key.
    lPosChannelKey = PosChannelKey(DEFAULT_POS, DEFAULT_CHANNEL);

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lRAC_SINBKKPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);
    FacBomManager::linkWithParent (lSINBKKDatePeriod, lRAC_SINBKKPosChannel);

    // Set the yield time-period primary key.
    const TimePeriodKey lYieldTimePeriodKey (lTimeRangeStart,
                                             lTimeRangeEnd);

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINBKKYieldTimePeriod =
      FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
    FacBomManager::addToListAndMap (lRAC_SINBKKPosChannel,
                                    lSINBKKYieldTimePeriod);
    FacBomManager::linkWithParent (lRAC_SINBKKPosChannel,
                                   lSINBKKYieldTimePeriod);

    // Generate the YieldRule
    const YieldFeaturesKey lYieldFeaturesKey (TRIP_TYPE_ONE_WAY,
                                              CABIN_Y);

```

```
    // Create the YieldFeaturesKey and link it to the TimePeriod object.
    YieldFeatures& lSINBKKYieldFeatures =
      FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
    FacBomManager::addToListAndMap (lSINBKKYieldTimePeriod,
                                    lSINBKKYieldFeatures);
    FacBomManager::linkWithParent (lSINBKKYieldTimePeriod,
                                   lSINBKKYieldFeatures);

    // Generate Segment Features and link them to their YieldRule.
    // Use the same key as previously.

    // Create the AirlineClassListKey and link it to the YieldFeatures object.
    AirlineClassList& lRAC_SQAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lSQAirlineYClassListKey);
    lRAC_SQAirlineYClassList.setYield(700);
    FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
                                    lRAC_SQAirlineYClassList);
    FacBomManager::linkWithParent (lSINBKKYieldFeatures,
                                   lRAC_SQAirlineYClassList);

    AirlineClassList& lRAC_SQAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQAirlineMClassListKey);
    lRAC_SQAirlineMClassList.setYield(500);
    FacBomManager::addToListAndMap (lSINBKKYieldFeatures,
                                    lRAC_SQAirlineMClassList);
    FacBomManager::linkWithParent (lSINBKKYieldFeatures,
                                   lRAC_SQAirlineMClassList);

    /*==========================================================================
    ========*/

    // Use the same airport pair, and date period for adding CX BKK-HKG yields.

    // Set the point-of-sale-channel primary key.
    // Use the same as previously.

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lRAC_BKKHKGPosChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);
    FacBomManager::linkWithParent (lBKKHKGDatePeriod, lRAC_BKKHKGPosChannel);

    // Set the yield time-period primary key.
    // Use the same as previously.

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lBKKHKGYieldTimePeriod =
      FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
    FacBomManager::addToListAndMap (lRAC_BKKHKGPosChannel,
                                    lBKKHKGYieldTimePeriod);
    FacBomManager::linkWithParent (lRAC_BKKHKGPosChannel,
                                   lBKKHKGYieldTimePeriod);

    // Generate the YieldRule
    // Use the same key as previously.

    // Create the YieldFeaturesKey and link it to the TimePeriod object.
    YieldFeatures& lBKKHKGYieldFeatures =
      FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
    FacBomManager::addToListAndMap (lBKKHKGYieldTimePeriod,
                                    lBKKHKGYieldFeatures);
    FacBomManager::linkWithParent (lBKKHKGYieldTimePeriod,
                                   lBKKHKGYieldFeatures);

    // Generate Segment Features and link them to their YieldRule.
    // Use the same key as previously.

```

```
    // Create the AirlineClassListKey and link it to the YieldFeatures object.
    AirlineClassList& lRAC_CXAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lCXAirlineYClassListKey);
    lRAC_CXAirlineYClassList.setYield(700);
    FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
                                    lRAC_CXAirlineYClassList);
    FacBomManager::linkWithParent (lBKKHKGYieldFeatures,
                                   lRAC_CXAirlineYClassList);

    AirlineClassList& lRAC_CXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lCXAirlineMClassListKey);
    lRAC_CXAirlineMClassList.setYield(500);
    FacBomManager::addToListAndMap (lBKKHKGYieldFeatures,
                                    lRAC_CXAirlineMClassList);
    FacBomManager::linkWithParent (lBKKHKGYieldFeatures,
                                   lRAC_CXAirlineMClassList);

    /*==========================================================================
========*/

    // Use the same airport pair, and date period for SQ-CX SIN-HKG

    // Set the point-of-sale-channel primary key.
    // Use the same as previously.

    // Create the PositionKey object and link it to the AirportPair object.
    PosChannel& lRAC_SINHKGChannel =
      FacBom<PosChannel>::instance().create (lPosChannelKey);
    FacBomManager::addToListAndMap (lSINHKGDatePeriod, lRAC_SINHKGChannel);
    FacBomManager::linkWithParent (lSINHKGDatePeriod, lRAC_SINHKGChannel);

    // Set the yield time-period primary key.
    // Use the same as previously.

    // Create the TimePeriodKey and link it to the DatePeriod object.
    TimePeriod& lSINHKGYieldTimePeriod =
      FacBom<TimePeriod>::instance().create (lYieldTimePeriodKey);
    FacBomManager::addToListAndMap (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);
    FacBomManager::linkWithParent (lRAC_SINHKGChannel, lSINHKGYieldTimePeriod);

    // Generate the YieldRule
    // Use the same key as previously.

    // Create the YieldFeaturesKey and link it to the TimePeriod object.
    YieldFeatures& lSINHKGYieldFeatures =
      FacBom<YieldFeatures>::instance().create (lYieldFeaturesKey);
    FacBomManager::addToListAndMap (lSINHKGYieldTimePeriod,
                                    lSINHKGYieldFeatures);
    FacBomManager::linkWithParent (lSINHKGYieldTimePeriod,
                                   lSINHKGYieldFeatures);

    // Generate Segment Features and link them to their YieldRule.
    // Use the same key as previously

    // Create the AirlineClassListKey and link it to the YieldFeatures object.
    AirlineClassList& lRAC_SQ_CXAirlineYClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineYClassListKey);
    lRAC_SQ_CXAirlineYClassList.setYield(1200);
    FacBomManager::addToListAndMap (lSINHKGYieldFeatures,
                                    lRAC_SQ_CXAirlineYClassList);
    FacBomManager::linkWithParent (lSINHKGYieldFeatures,
                                   lRAC_SQ_CXAirlineYClassList);

    AirlineClassList& lRAC_SQ_CXAirlineMClassList =
      FacBom<AirlineClassList>::instance().create (lSQ_CXAirlineMClassListKey);
    lRAC_SQ_CXAirlineMClassList.setYield(850);
```

```
    FacBomManager::addToListAndMap (lSINHKGYieldFeatures,
                                    lRAC_SQ_CXAirlineMClassList);
    FacBomManager::linkWithParent (lSINHKGYieldFeatures,
                                   lRAC_SQ_CXAirlineMClassList);

  }

}
```

33.539 stdair/command/CmdBomManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/basic/SampleType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class stdair::CmdBomManager

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.540 stdair/command/CmdBomManager.hpp

```
00001 #ifndef __STDAIR_CMD_CMDBOMMANAGER_HPP
00002 #define __STDAIR_CMD_CMDBOMMANAGER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/stdair_inventory_types.hpp>
00011 #include <stdair/basic/SampleType.hpp>
00012 #include <stdair/bom/TravelSolutionTypes.hpp>
00013 #include <stdair/command/CmdAbstract.hpp>
00014
00015 namespace stdair {
00016
00018   class BomRoot;
00019   struct BookingRequestStruct;
00020
00025   class CmdBomManager : public CmdAbstract {
00026     //
00027     friend class STDAIR_Service;
00028   private:
00029
00030     // ///////////////// BOM initialisation support methods /////////////////
00042     static void buildSampleBom (BomRoot&);
00043
00055     static void buildSampleInventorySchedule (BomRoot&);
00056
00067     static void buildSampleInventoryScheduleForFareFamilies (BomRoot&);
00068
00082     static void buildCompleteDummyInventory (BomRoot&);
00083
00097     static void buildCompleteDummyInventoryForFareFamilies (BomRoot&);
00098
00099
00117     static void buildDummyInventory (BomRoot&, const CabinCapacity_T&);
00118
00131     static void buildDummyLegSegmentAccesses (BomRoot&);
00132
00140     static void buildSamplePricing (BomRoot&);
00141
00150     static void buildSamplePricingForFareFamilies (BomRoot&);
00151
00166     static void buildSampleTravelSolutionForPricing (TravelSolutionList_T&);
00167
00185     static void buildSampleTravelSolutions (TravelSolutionList_T&);
00186
00203     static BookingRequestStruct buildSampleBookingRequest();
00204
00221     static BookingRequestStruct buildSampleBookingRequestForCRS();
00222
00233     static void buildPartnershipsSampleInventoryAndRM (BomRoot&);
00234
00242     static void buildPartnershipsSamplePricing (BomRoot&);
00243
00244   };
00245 }
00246 #endif // ___STDAIR_CMD_CMDBOMMANAGER_HPP
```

33.541 stdair/command/CmdBomSerialiser.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <boost/archive/text_iarchive.hpp>
#include <boost/archive/text_oarchive.hpp>
#include <boost/serialization/list.hpp>
#include <boost/serialization/map.hpp>
#include <boost/serialization/access.hpp>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/command/CmdBomSerialiser.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template<class Archive , class BOM_OBJECT1 , class BOM_OBJECT2 >
  void stdair::serialiseHelper (BOM_OBJECT1 &ioObject1, Archive &ioArchive, const unsigned int iFileVersion)
- template void stdair::BomRoot::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::BomRoot::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::Inventory::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::Inventory::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::FlightDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::FlightDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

- template void stdair::SegmentDate::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentDate::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)
- template void stdair::SegmentCabin::serialize< ba::text_oarchive > (ba::text_oarchive &, unsigned int)
- template void stdair::SegmentCabin::serialize< ba::text_iarchive > (ba::text_iarchive &, unsigned int)

33.542 stdair/command/CmdBomSerialiser.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 // Boost.Serialization
00008 #include <boost/archive/text_iarchive.hpp>
00009 #include <boost/archive/text_oarchive.hpp>
00010 #include <boost/serialization/list.hpp>
00011 #include <boost/serialization/map.hpp>
00012 #include <boost/serialization/access.hpp>
00013 // StdAir
00014 #include <stdair/basic/BasConst_General.hpp>
00015 #include <stdair/basic/BasConst_Inventory.hpp>
00016 #include <stdair/bom/BomRoot.hpp>
00017 #include <stdair/bom/Inventory.hpp>
00018 #include <stdair/bom/FlightDate.hpp>
00019 #include <stdair/bom/SegmentDate.hpp>
00020 #include <stdair/bom/SegmentCabin.hpp>
00021 #include <stdair/bom/FareFamily.hpp>
00022 #include <stdair/bom/LegDate.hpp>
00023 #include <stdair/bom/LegCabin.hpp>
00024 #include <stdair/bom/Bucket.hpp>
00025 #include <stdair/factory/FacBomManager.hpp>
00026 #include <stdair/factory/FacBom.hpp>
00027 #include <stdair/command/CmdBomSerialiser.hpp>
00028 #include <stdair/service/Logger.hpp>
00029
00030 namespace stdair {
00031
00032   // ////////////////////////////////////////////////////////////////////
00033   template <class Archive, class BOM_OBJECT1, class BOM_OBJECT2>
00034   void serialiseHelper (BOM_OBJECT1& ioObject1, Archive& ioArchive,
00035                         const unsigned int iFileVersion) {
00036
00050     BomHolder<BOM_OBJECT2>* lBomHolder_ptr =
00051       FacBomManager::getBomHolderPtr<BOM_OBJECT2> (ioObject1);
00052
00053     if (lBomHolder_ptr == NULL) {
00054       lBomHolder_ptr = &FacBomManager::addBomHolder<BOM_OBJECT2> (ioObject1);
00055     }
00056     assert (lBomHolder_ptr != NULL);
00057
00061     //ioArchive.register_type (static_cast<Inventory*> (NULL));
00062     ioArchive & lBomHolder_ptr->_bomList;
00063     ioArchive & lBomHolder_ptr->_bomMap;
00064
00071     typedef typename BomHolder<BOM_OBJECT2>::BomList_T BomList_T;
00072     const BomList_T& lBomList = lBomHolder_ptr->_bomList;
00073     for (typename BomList_T::const_iterator itObject = lBomList.begin();
00074          itObject != lBomList.end(); ++itObject) {
00075       BOM_OBJECT2* lObject2_ptr = *itObject;
00076       assert (lObject2_ptr != NULL);
00077
00078       if (lObject2_ptr->getParent() == NULL) {
00084         FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00085       }
00086     }
00087
00096     typedef typename BomHolder<BOM_OBJECT2>::BomMap_T BomMap_T;
00097     const BomMap_T& lBomMap = lBomHolder_ptr->_bomMap;
00098     if (lBomList.empty() == true && lBomMap.empty() == false) {
00099
00100       for (typename BomMap_T::const_iterator itObject = lBomMap.begin();
```

```
00101           itObject != lBomMap.end(); ++itObject) {
00102        BOM_OBJECT2* lObject2_ptr = itObject->second;
00103        assert (lObject2_ptr != NULL);
00104
00105        if (lObject2_ptr->getParent() == NULL) {
00111          FacBomManager::linkWithParent (ioObject1, *lObject2_ptr);
00112        }
00113      }
00114    }
00115  }
00116
00117  // ////////////////////////////////////////////////////////////////////
00118  void BomRoot::serialisationImplementationExport() const {
00119    std::ostringstream oStr;
00120    boost::archive::text_oarchive oa (oStr);
00121    oa << *this;
00122  }
00123
00124  // ////////////////////////////////////////////////////////////////////
00125  void BomRoot::serialisationImplementationImport() {
00126    std::istringstream iStr;
00127    boost::archive::text_iarchive ia (iStr);
00128    ia >> *this;
00129  }
00130
00131  // ////////////////////////////////////////////////////////////////////
00132  template<class Archive>
00133  void BomRoot::serialize (Archive& ioArchive,
00134                           const unsigned int iFileVersion) {
00135    // Serialise the key (by default, equal to " -- ROOT -- ")
00136    ioArchive & _key;
00137
00138    // Serialise the children of the BomRoot object, i.e., the
00139    // Inventory children
00140    stdair::serialiseHelper<Archive, BomRoot, Inventory> (*this, ioArchive,
00141                                                          iFileVersion);
00142  }
00143
00144  // ////////////////////////////////////////////////////////////////////
00145  void Inventory::serialisationImplementationExport() const {
00146    std::ostringstream oStr;
00147    boost::archive::text_oarchive oa (oStr);
00148    oa << *this;
00149  }
00150
00151  // ////////////////////////////////////////////////////////////////////
00152  void Inventory::serialisationImplementationImport() {
00153    std::istringstream iStr;
00154    boost::archive::text_iarchive ia (iStr);
00155    ia >> *this;
00156  }
00157
00158  // ////////////////////////////////////////////////////////////////////
00159  template<class Archive>
00160  void Inventory::serialize (Archive& ioArchive,
00161                             const unsigned int iFileVersion) {
00162    // Serialise the key (airline code)
00163    ioArchive & _key;
00164
00165    // Serialise the children of the Inventory object, i.e., the
00166    // FlightDate children
00167    stdair::serialiseHelper<Archive, Inventory, FlightDate> (*this, ioArchive,
00168                                                             iFileVersion);
00169  }
00170
00171  // ////////////////////////////////////////////////////////////////////
00172  void FlightDate::serialisationImplementationExport() const {
```

```
  std::ostringstream oStr;
  boost::archive::text_oarchive oa (oStr);
  oa << *this;
}

// ////////////////////////////////////////////////////////////////////
void FlightDate::serialisationImplementationImport() {
  std::istringstream iStr;
  boost::archive::text_iarchive ia (iStr);
  ia >> *this;
}

// ////////////////////////////////////////////////////////////////////
template<class Archive>
void FlightDate::serialize (Archive& ioArchive,
                            const unsigned int iFileVersion) {
  ioArchive & _key;
}

// ////////////////////////////////////////////////////////////////////
void SegmentDate::serialisationImplementationExport() const {
  std::ostringstream oStr;
  boost::archive::text_oarchive oa (oStr);
  oa << *this;
}

// ////////////////////////////////////////////////////////////////////
void SegmentDate::serialisationImplementationImport() {
  std::istringstream iStr;
  boost::archive::text_iarchive ia (iStr);
  ia >> *this;
}

// ////////////////////////////////////////////////////////////////////
template<class Archive>
void SegmentDate::serialize (Archive& ioArchive,
                            const unsigned int iFileVersion) {
  ioArchive & _key;
}

// ////////////////////////////////////////////////////////////////////
void SegmentCabin::serialisationImplementationExport() const {
  std::ostringstream oStr;
  boost::archive::text_oarchive oa (oStr);
  oa << *this;
}

// ////////////////////////////////////////////////////////////////////
void SegmentCabin::serialisationImplementationImport() {
  std::istringstream iStr;
  boost::archive::text_iarchive ia (iStr);
  ia >> *this;
}

// ////////////////////////////////////////////////////////////////////
template<class Archive>
void SegmentCabin::serialize (Archive& ioArchive,
                              const unsigned int iFileVersion) {
  ioArchive & _key;
}

// ////////////////////////////////////////////////////////////////////
// Explicit template instantiations
namespace ba = boost::archive;
template void BomRoot::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                     unsigned int);
template void BomRoot::serialize<ba::text_iarchive> (ba::text_iarchive&,
```

```
                                                        unsigned int);
  template void Inventory::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                         unsigned int);
  template void Inventory::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                         unsigned int);
  template void FlightDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                          unsigned int);
  template void FlightDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                          unsigned int);
  template void SegmentDate::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                           unsigned int);
  template void SegmentDate::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                           unsigned int);
  template void SegmentCabin::serialize<ba::text_oarchive> (ba::text_oarchive&,
                                                            unsigned int);
  template void SegmentCabin::serialize<ba::text_iarchive> (ba::text_iarchive&,
                                                            unsigned int);
  // ////////////////////////////////////////////////////////////////////
}
```

33.543 stdair/command/CmdBomSerialiser.hpp File Reference

```
#include <iosfwd>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class stdair::CmdBomSerialiser

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.544 stdair/command/CmdBomSerialiser.hpp

```
#ifndef __STDAIR_CMD_CMDBOMSERIALISER_HPP
#define __STDAIR_CMD_CMDBOMSERIALISER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/command/CmdAbstract.hpp>

namespace stdair {

  class BomRoot;
  struct BookingRequestStruct;


  class CmdBomSerialiser : public CmdAbstract {
  public:

    // /////////////// BOM serialisation support methods /////////////////
    //
  };
}
#endif // ___STDAIR_CMD_CMDBOMSERIALISER_HPP
```

33.545 stdair/command/CmdCloneBomManager.cpp File Reference

33.546 stdair/command/CmdCloneBomManager.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 #include <sstream>
00011 // StdAir
00012 #include <stdair/factory/FacBomManager.hpp>
00013 #include <stdair/factory/FacCloneBom.hpp>
00014 #include <stdair/command/CmdCloneBomManager.hpp>
00015 #include <stdair/service/Logger.hpp>
00016 #include <stdair/bom/BomRetriever.hpp>
00017
00018 namespace stdair {
00019
00020   // ////////////////////////////////////////////////////////////////////
00021   void CmdCloneBomManager::cloneBomRoot (const BomRoot& iBomRoot,
00022                                          BomRoot& ioCloneBomRoot) {
00023
00029     // Check whether there are Inventory objects
00030     const bool hasInventoryList = BomManager::hasList<Inventory> (iBomRoot);
00031     if (hasInventoryList == true) {
00032
00033       // Browse the inventories
00034       const InventoryList_T& lInventoryList =
00035         BomManager::getList<Inventory> (iBomRoot);
00036       for (InventoryList_T::const_iterator itInv = lInventoryList.begin();
00037            itInv != lInventoryList.end(); ++itInv) {
00038         const Inventory* lInv_ptr = *itInv;
00039         assert (lInv_ptr != NULL);
00040
00041         // Clone the current inventory
00042         Inventory& lCloneInventory = cloneInventory (*lInv_ptr, ioCloneBomRoot);
00043         FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneInventory);
00044         FacBomManager::linkWithParent (ioCloneBomRoot, lCloneInventory);
00045       }
00046     }
00047
00048     // Check whether there are Airport Pair objects
00049     const bool hastAirportPairList =
00050       BomManager::hasList<AirportPair> (iBomRoot);
00051     if (hastAirportPairList == true) {
00052
00053       // Browse the airport pairs
00054       const AirportPairList_T& lAirportPairList =
00055         BomManager::getList<AirportPair> (iBomRoot);
00056       for (AirportPairList_T::const_iterator itAirportPair =
00057              lAirportPairList.begin();
00058            itAirportPair != lAirportPairList.end(); ++itAirportPair) {
00059         const AirportPair* lAirportPair_ptr = *itAirportPair;
00060         assert (lAirportPair_ptr != NULL);
00061
00062         // Clone the current airport pair
00063         AirportPair& lCloneAirportPair = cloneAirportPair (*lAirportPair_ptr);
00064         FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirportPair);
00065         FacBomManager::linkWithParent (ioCloneBomRoot, lCloneAirportPair);
00066       }
00067     }
00068   }
00069
00070   // ////////////////////////////////////////////////////////////////////
00071   Inventory& CmdCloneBomManager::cloneInventory (const Inventory& iInventory,
00072                                                  BomRoot& ioCloneBomRoot) {
00073
```

```
    Inventory& lCloneInventory =
      FacCloneBom<Inventory>::instance().clone (iInventory);

    // Check whether there are FlightDate objects
    const bool hasFlighDateList = BomManager::hasList<FlightDate> (iInventory);
    if (hasFlighDateList == true) {
      // Browse the flight-dates
      const FlightDateList_T& lFlightDateList =
        BomManager::getList<FlightDate> (iInventory);
      for (FlightDateList_T::const_iterator itFD = lFlightDateList.begin();
           itFD != lFlightDateList.end(); ++itFD) {
        const FlightDate* lFD_ptr = *itFD;
        assert (lFD_ptr != NULL);

        // Clone the current flight-date
        FlightDate& lCloneFD = cloneFlightDate (*lFD_ptr);
        FacBomManager::addToListAndMap (lCloneInventory, lCloneFD);
        FacBomManager::linkWithParent (lCloneInventory, lCloneFD);
      }
    }

    // Check if the inventory contains a list of partners
    const bool hasPartnerList = BomManager::hasList<Inventory> (iInventory);
    if (hasPartnerList == true) {

      // Browse the partner's inventories
      const InventoryList_T& lPartnerInventoryList =
        BomManager::getList<Inventory> (iInventory);

      for (InventoryList_T::const_iterator itInv =
             lPartnerInventoryList.begin();
           itInv != lPartnerInventoryList.end(); ++itInv) {
        const Inventory* lInv_ptr = *itInv;
        assert (lInv_ptr != NULL);

        // Clone the current partnership inventory
        Inventory& lClonePartnerInventory = cloneInventory (*lInv_ptr,
                                                            ioCloneBomRoot);
        FacBomManager::addToListAndMap (lCloneInventory,
                                        lClonePartnerInventory);
        FacBomManager::linkWithParent (lCloneInventory,
                                       lClonePartnerInventory);
      }
    }

    // Check whether there are O&D date objects
    const bool hasOnDList = BomManager::hasList<OnDDate> (iInventory);
    if (hasOnDList == true){

      //Browse the O&Ds
      const OnDDateList_T& lOnDDateList =
        BomManager::getList<OnDDate> (iInventory);

      for (OnDDateList_T::const_iterator itOnD = lOnDDateList.begin();
           itOnD != lOnDDateList.end(); ++itOnD) {
        const OnDDate* lOnDDate_ptr = *itOnD;
        assert (lOnDDate_ptr != NULL);

        // Clone the current O&D date
        OnDDate& lCloneOnDDate = cloneOnDDate (*lOnDDate_ptr);
        FacBomManager::addToListAndMap (lCloneInventory, lCloneOnDDate);
        FacBomManager::linkWithParent (lCloneInventory, lCloneOnDDate);
      }
    }

    // Check whether there are Flight Period objects
    const bool hasFlightPeriodList =
```

```
      BomManager::hasList<FlightPeriod> (iInventory);
    if (hasFlightPeriodList == true) {

      // Browse the flight-periods
      const FlightPeriodList_T& lFlightPeriodList =
        BomManager::getList<FlightPeriod> (iInventory);
      for (FlightPeriodList_T::const_iterator itFlightPeriod =
             lFlightPeriodList.begin();
           itFlightPeriod != lFlightPeriodList.end(); ++itFlightPeriod) {
        const FlightPeriod* lFlightPeriod_ptr = *itFlightPeriod;
        assert (lFlightPeriod_ptr != NULL);

        // Clone the current flight period
        FlightPeriod& lCloneFlightPeriod = cloneFlightPeriod (*lFlightPeriod_ptr);
        FacBomManager::addToListAndMap (lCloneInventory, lCloneFlightPeriod);
        FacBomManager::linkWithParent (lCloneInventory, lCloneFlightPeriod);
      }
    }

    // Check whether there is an airline feature object
    const AirlineFeature* lAirlineFeature_ptr =
      BomManager::getObjectPtr<AirlineFeature,Inventory> (iInventory,
                                                          iInventory.getAirlineCode());
    if (lAirlineFeature_ptr != NULL) {
      // Clone the current airline feature object
      AirlineFeature& lCloneAirlineFeature =
        cloneAirlineFeature (*lAirlineFeature_ptr);
      FacBomManager::setAirlineFeature (lCloneInventory, lCloneAirlineFeature);
      FacBomManager::linkWithParent (lCloneInventory, lCloneAirlineFeature);
      // Link the airline feature object with the top of the BOM tree
      FacBomManager::addToListAndMap (ioCloneBomRoot, lCloneAirlineFeature);
    }

    return lCloneInventory;
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineFeature& CmdCloneBomManager::
  cloneAirlineFeature (const AirlineFeature& iAirlineFeature) {

    AirlineFeature& lCloneAirlineFeature =
      FacCloneBom<AirlineFeature>::instance().clone (iAirlineFeature);

    return lCloneAirlineFeature;
  }


  // ////////////////////////////////////////////////////////////////////
  OnDDate& CmdCloneBomManager::cloneOnDDate (const OnDDate& iOnDDate) {

    OnDDate& lCloneOnDDate =
      FacCloneBom<OnDDate>::instance().clone (iOnDDate);

    return lCloneOnDDate;
  }

  // ////////////////////////////////////////////////////////////////////
  FlightDate& CmdCloneBomManager::
  cloneFlightDate (const FlightDate& iFlightDate) {

    FlightDate& lCloneFlightDate =
      FacCloneBom<FlightDate>::instance().clone (iFlightDate);

    // Check whether there are LegDate objects
    const bool hasLegDateList = BomManager::hasList<LegDate> (iFlightDate);
```

```
    if (hasLegDateList == true) {

      // Browse the leg-dates
      const LegDateList_T& lLegDateList =
        BomManager::getList<LegDate> (iFlightDate);
      for (LegDateList_T::const_iterator itLD = lLegDateList.begin();
           itLD != lLegDateList.end(); ++itLD) {
        const LegDate* lLD_ptr = *itLD;
        assert (lLD_ptr != NULL);

        // Clone the current leg-date
        LegDate& lCloneLegDate = cloneLegDate (*lLD_ptr);
        FacBomManager::addToListAndMap (lCloneFlightDate, lCloneLegDate);
        FacBomManager::linkWithParent (lCloneFlightDate, lCloneLegDate);
      }
    }

    // Check whether there are SegmentDate objects
    const bool hasSegmentDateList =
      BomManager::hasList<SegmentDate> (iFlightDate);
    if (hasSegmentDateList == true) {

      // Browse the segment-dates
      const SegmentDateList_T& lSegmentDateList =
        BomManager::getList<SegmentDate> (iFlightDate);
      for (SegmentDateList_T::const_iterator itSD = lSegmentDateList.begin();
           itSD != lSegmentDateList.end(); ++itSD) {
        const SegmentDate* lSD_ptr = *itSD;
        assert (lSD_ptr != NULL);

        // Clone the current segment-date
        SegmentDate& lCloneSegmentDate = cloneSegmentDate (*lSD_ptr);
        FacBomManager::addToListAndMap (lCloneFlightDate, lCloneSegmentDate);
        FacBomManager::linkWithParent (lCloneFlightDate, lCloneSegmentDate);

      }
    }

    return lCloneFlightDate;
  }

  // ////////////////////////////////////////////////////////////////////
  LegDate& CmdCloneBomManager::cloneLegDate (const LegDate& iLegDate) {

    LegDate& lCloneLegDate =
      FacCloneBom<LegDate>::instance().clone (iLegDate);

    // Check whether there are LegCabin objects
    const bool hasLegCabinList = BomManager::hasList<LegCabin> (iLegDate);
    if (hasLegCabinList == true) {
      // Browse the leg-cabins
      const LegCabinList_T& lLegCabinList =
        BomManager::getList<LegCabin> (iLegDate);
      for (LegCabinList_T::const_iterator itLC = lLegCabinList.begin();
           itLC != lLegCabinList.end(); ++itLC) {
        const LegCabin* lLC_ptr = *itLC;
        assert (lLC_ptr != NULL);

        // Clone the current leg-cabin
        LegCabin& lCloneLegCabin = cloneLegCabin (*lLC_ptr);
        FacBomManager::addToListAndMap (lCloneLegDate, lCloneLegCabin);
        FacBomManager::linkWithParent (lCloneLegDate, lCloneLegCabin);
      }
    }

    return lCloneLegDate;
  }
```

```

// ////////////////////////////////////////////////////////////////////
LegCabin& CmdCloneBomManager::cloneLegCabin (const LegCabin& iLegCabin) {

  LegCabin& lCloneLegCabin =
    FacCloneBom<LegCabin>::instance().clone (iLegCabin);

  // Check whether there are Bucket objects
  const bool hasBucketList = BomManager::hasList<Bucket> (iLegCabin);
  if (hasBucketList == true) {
    // Browse the buckets
    const BucketList_T& lBucketList =
      BomManager::getList<Bucket> (iLegCabin);
    for (BucketList_T::const_iterator itBucket = lBucketList.begin();
         itBucket != lBucketList.end(); ++itBucket) {
      const Bucket* lBucket_ptr = *itBucket;
      assert (lBucket_ptr != NULL);

      // Clone the current bucket
      Bucket& lCloneBucket = cloneBucket (*lBucket_ptr);
      FacBomManager::addToListAndMap (lCloneLegCabin, lCloneBucket);
      FacBomManager::linkWithParent (lCloneLegCabin, lCloneBucket);
    }
  }

  return lCloneLegCabin;
}

// ////////////////////////////////////////////////////////////////////
Bucket& CmdCloneBomManager::cloneBucket (const Bucket& iBucket) {

  Bucket& lCloneBucket =
    FacCloneBom<Bucket>::instance().clone (iBucket);

  return lCloneBucket;
}

// ////////////////////////////////////////////////////////////////////
SegmentDate& CmdCloneBomManager::
cloneSegmentDate (const SegmentDate& iSegmentDate) {

  SegmentDate& lCloneSegmentDate =
    FacCloneBom<SegmentDate>::instance().clone (iSegmentDate);

  // Check whether there are SegmentCabin objects
  const bool hasSegmentCabinList =
    BomManager::hasList<SegmentCabin> (iSegmentDate);
  if (hasSegmentCabinList == true) {
    // Browse the segment-cabins
    const SegmentCabinList_T& lSegmentCabinList =
      BomManager::getList<SegmentCabin> (iSegmentDate);
    for (SegmentCabinList_T::const_iterator itSC = lSegmentCabinList.begin();
         itSC != lSegmentCabinList.end(); ++itSC) {
      const SegmentCabin* lSC_ptr = *itSC;
      assert (lSC_ptr != NULL);

      // Clone the current segment-cabin
      SegmentCabin& lCloneSegmentCabin = cloneSegmentCabin (*lSC_ptr);
      FacBomManager::addToListAndMap (lCloneSegmentDate, lCloneSegmentCabin);
      FacBomManager::linkWithParent (lCloneSegmentDate, lCloneSegmentCabin);

      linkBookingClassesWithSegment (lCloneSegmentDate,
                                     lCloneSegmentCabin);

    }
  }
  return lCloneSegmentDate;
```

```
}

// ////////////////////////////////////////////////////////////////////
void CmdCloneBomManager::
linkBookingClassesWithSegment (SegmentDate& iCloneSegmentDate,
                               SegmentCabin& iCloneSegmentCabin) {

  // Browse the fare families to link the booking-classes to the
  // segment-cabin and to the segment-date
  const bool hasFareFamilyList =
    BomManager::hasList<FareFamily> (iCloneSegmentCabin);
  if (hasFareFamilyList == true) {
    const FareFamilyList_T& lCloneFFList =
      BomManager::getList<FareFamily> (iCloneSegmentCabin);
    for (FareFamilyList_T::const_iterator itCloneFF = lCloneFFList.begin();
         itCloneFF != lCloneFFList.end(); ++itCloneFF) {
      const FareFamily* lCloneFF_ptr = *itCloneFF;
      assert (lCloneFF_ptr != NULL);

      // Browse the list of booking classes
      const bool hasBookingClasslist =
        BomManager::hasList<BookingClass> (*lCloneFF_ptr);
      if (hasBookingClasslist == true) {
        const BookingClassList_T& lCloneBCList =
          BomManager::getList<BookingClass> (*lCloneFF_ptr);
        for (BookingClassList_T::const_iterator itCloneBC =
               lCloneBCList.begin();
             itCloneBC != lCloneBCList.end(); ++itCloneBC) {
          const BookingClass* lCloneBC_ptr = *itCloneBC;
          assert (lCloneBC_ptr != NULL);

          // Link the booking-class to the segment-cabin
          stdair::FacBomManager::addToListAndMap (iCloneSegmentCabin,
                                                  *lCloneBC_ptr);

          // Link the booking-class to the segment-date
          stdair::FacBomManager::addToListAndMap (iCloneSegmentDate,
                                                  *lCloneBC_ptr);
        }
      }
    }
  }
}

// ////////////////////////////////////////////////////////////////////
SegmentCabin& CmdCloneBomManager::
cloneSegmentCabin (const SegmentCabin& iSegmentCabin) {

  SegmentCabin& lCloneSegmentCabin =
    FacCloneBom<SegmentCabin>::instance().clone (iSegmentCabin);

  // Check whether there are fare family objects
  const bool hasFareFamilyList =
    BomManager::hasList<FareFamily> (iSegmentCabin);
  if (hasFareFamilyList == true) {
    // Browse the fare families
    const FareFamilyList_T& lFareFamilyList =
      BomManager::getList<FareFamily> (iSegmentCabin);
    for (FareFamilyList_T::const_iterator itFF = lFareFamilyList.begin();
         itFF != lFareFamilyList.end(); ++itFF) {
      const FareFamily* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Clone the current fare-family
      FareFamily& lCloneFareFamily = cloneFareFamily (*lFF_ptr);
      FacBomManager::addToListAndMap (lCloneSegmentCabin, lCloneFareFamily);
      FacBomManager::linkWithParent (lCloneSegmentCabin, lCloneFareFamily);
```

```
      }
    }

    return lCloneSegmentCabin;
  }

  // ////////////////////////////////////////////////////////////////////
  FareFamily& CmdCloneBomManager::
  cloneFareFamily (const FareFamily& iFareFamily) {
    FareFamily& lCloneFareFamily =
      FacCloneBom<FareFamily>::instance().clone (iFareFamily);

    // Check whether there are booking classes objects
    const bool hasBookingClassList =
      BomManager::hasList<BookingClass> (iFareFamily);
    if (hasBookingClassList == true) {
      // Browse the list of booking classes
      const BookingClassList_T& lBookingClassList =
        BomManager::getList<BookingClass> (iFareFamily);
      for (BookingClassList_T::const_iterator itBookingClass =
             lBookingClassList.begin();
           itBookingClass != lBookingClassList.end(); ++itBookingClass) {
        const BookingClass* lBC_ptr = *itBookingClass;
        assert (lBC_ptr != NULL);

        // Clone the current booking class
        BookingClass& lCloneBookingClass = cloneBookingClass (*lBC_ptr);
        FacBomManager::addToListAndMap (lCloneFareFamily, lCloneBookingClass);
        FacBomManager::linkWithParent (lCloneFareFamily, lCloneBookingClass);
      }
    }

    return lCloneFareFamily;
  }

  // ////////////////////////////////////////////////////////////////////
  BookingClass& CmdCloneBomManager::
  cloneBookingClass (const BookingClass& iBookingClass) {

    BookingClass& lCloneBookingClass =
      FacCloneBom<BookingClass>::instance().clone (iBookingClass);

    return lCloneBookingClass;
  }

  // ////////////////////////////////////////////////////////////////////
  AirportPair& CmdCloneBomManager::
  cloneAirportPair (const AirportPair& iAirportPair) {

    AirportPair& lCloneAirportPair =
      FacCloneBom<AirportPair>::instance().clone (iAirportPair);

    // Check whether there are date-period objects
    const bool hasDatePeriodList =
      BomManager::hasList<DatePeriod> (iAirportPair);
    if (hasDatePeriodList == true) {
      // Browse the date-periods
      const DatePeriodList_T& lDatePeriodList =
        BomManager::getList<DatePeriod> (iAirportPair);
      for (DatePeriodList_T::const_iterator itDatePeriod =
             lDatePeriodList.begin();
           itDatePeriod != lDatePeriodList.end(); ++itDatePeriod) {
        const DatePeriod* lDatePeriod_ptr = *itDatePeriod;
        assert (lDatePeriod_ptr != NULL);

        // Clone the current date-period
        DatePeriod& lCloneDatePeriod = cloneDatePeriod (*lDatePeriod_ptr);
```

```
        FacBomManager::addToListAndMap (lCloneAirportPair, lCloneDatePeriod);
        FacBomManager::linkWithParent (lCloneAirportPair, lCloneDatePeriod);
      }
    }

    return lCloneAirportPair;
  }

  // ////////////////////////////////////////////////////////////////////
  DatePeriod& CmdCloneBomManager::
  cloneDatePeriod (const DatePeriod& iDatePeriod) {

    DatePeriod& lCloneDatePeriod =
      FacCloneBom<DatePeriod>::instance().clone (iDatePeriod);

    // Check whether there are pos-channel objects
    const bool hasPosChannelList =
      BomManager::hasList<PosChannel> (iDatePeriod);
    if (hasPosChannelList == true) {
      // Browse the pos-channels
      const PosChannelList_T& lPosChannelList =
        BomManager::getList<PosChannel> (iDatePeriod);
      for (PosChannelList_T::const_iterator itPosChannel =
             lPosChannelList.begin();
           itPosChannel != lPosChannelList.end(); ++itPosChannel) {
        const PosChannel* lPosChannel_ptr = *itPosChannel;
        assert (lPosChannel_ptr != NULL);

        // Clone the current pos-channel
        PosChannel& lClonePosChannel = clonePosChannel (*lPosChannel_ptr);
        FacBomManager::addToListAndMap (lCloneDatePeriod, lClonePosChannel);
        FacBomManager::linkWithParent (lCloneDatePeriod, lClonePosChannel);
      }
    }

    return lCloneDatePeriod;
  }


  // ////////////////////////////////////////////////////////////////////
  PosChannel& CmdCloneBomManager::
  clonePosChannel (const PosChannel& iPosChannel) {

    PosChannel& lClonePosChannel =
      FacCloneBom<PosChannel>::instance().clone (iPosChannel);

    // Check whether there are time-period objects
    const bool hasTimePeriodList =
      BomManager::hasList<TimePeriod> (iPosChannel);
    if (hasTimePeriodList == true) {
      // Browse the time-periods
      const TimePeriodList_T& lTimePeriodList =
        BomManager::getList<TimePeriod> (iPosChannel);
      for (TimePeriodList_T::const_iterator itTimePeriod =
             lTimePeriodList.begin();
           itTimePeriod != lTimePeriodList.end(); ++itTimePeriod) {
        const TimePeriod* lTimePeriod_ptr = *itTimePeriod;
        assert (lTimePeriod_ptr != NULL);

        // Clone the current time-period
        TimePeriod& lCloneTimePeriod = cloneTimePeriod (*lTimePeriod_ptr);
        FacBomManager::addToListAndMap (lClonePosChannel, lCloneTimePeriod);
        FacBomManager::linkWithParent (lClonePosChannel, lCloneTimePeriod);
      }
    }

    return lClonePosChannel;
```

```
}

// ////////////////////////////////////////////////////////////////////
TimePeriod& CmdCloneBomManager::
cloneTimePeriod (const TimePeriod& iTimePeriod) {

  TimePeriod& lCloneTimePeriod =
    FacCloneBom<TimePeriod>::instance().clone (iTimePeriod);

  // Check whether there are fare-feature objects
  const bool hasFareFeaturesList =
    BomManager::hasList<FareFeatures> (iTimePeriod);
  if (hasFareFeaturesList == true) {
    // Browse the fare-features
    const FareFeaturesList_T& lFareFeaturesList =
      BomManager::getList<FareFeatures> (iTimePeriod);
    for (FareFeaturesList_T::const_iterator itFF = lFareFeaturesList.begin();
         itFF != lFareFeaturesList.end(); ++itFF) {
      const FareFeatures* lFF_ptr = *itFF;
      assert (lFF_ptr != NULL);

      // Clone the current fare-feature
      FareFeatures& lCloneFareFeatures =
        cloneFeatures<FareFeatures> (*lFF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneFareFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneFareFeatures);
    }
  }

  // Check whether there are yield-feature objects
  const bool hasYieldFeaturesList =
    BomManager::hasList<YieldFeatures> (iTimePeriod);
  if (hasYieldFeaturesList == true) {
    // Browse the yield-features
    const YieldFeaturesList_T& lYieldFeaturesList =
      BomManager::getList<YieldFeatures> (iTimePeriod);
    for (YieldFeaturesList_T::const_iterator itYF =
           lYieldFeaturesList.begin();
         itYF != lYieldFeaturesList.end(); ++itYF) {
      const YieldFeatures* lYF_ptr = *itYF;
      assert (lYF_ptr != NULL);

      // Clone the current yield-feature
      YieldFeatures& lCloneYieldFeatures =
        cloneFeatures<YieldFeatures> (*lYF_ptr);
      FacBomManager::addToListAndMap (lCloneTimePeriod, lCloneYieldFeatures);
      FacBomManager::linkWithParent (lCloneTimePeriod, lCloneYieldFeatures);
    }
  }

  return lCloneTimePeriod;
}

// ////////////////////////////////////////////////////////////////////
template <typename FEATURE_TYPE>
FEATURE_TYPE& CmdCloneBomManager::
cloneFeatures (const FEATURE_TYPE& iFeatures) {

  FEATURE_TYPE& lCloneFeatures =
    FacCloneBom<FEATURE_TYPE>::instance().clone (iFeatures);

  // Check whether there are airline-class list objects
  const bool hasAirlineClassListList =
    BomManager::hasList<AirlineClassList> (iFeatures);
  if (hasAirlineClassListList == true) {
    // Browse the airline-class lists
    const AirlineClassListList_T& lAirlineClassList =
```

```
        BomManager::getList<AirlineClassList> (iFeatures);
      for (AirlineClassListList_T::const_iterator itACList =
             lAirlineClassList.begin();
           itACList != lAirlineClassList.end(); ++itACList) {
        const AirlineClassList* lACList_ptr = *itACList;
        assert (lACList_ptr != NULL);

        // Clone the current airline-class list
        AirlineClassList& lCloneAirlineClassList =
          cloneAirlineClassList (*lACList_ptr);
        FacBomManager::addToListAndMap (lCloneFeatures,
                                        lCloneAirlineClassList);
        FacBomManager::linkWithParent (lCloneFeatures,
                                       lCloneAirlineClassList);
      }
    }

    return lCloneFeatures;
  }

  // ////////////////////////////////////////////////////////////////////
  AirlineClassList& CmdCloneBomManager::
  cloneAirlineClassList (const AirlineClassList& iAirlineClassList) {

    AirlineClassList& lCloneAirlineClassList =
      FacCloneBom<AirlineClassList>::instance().clone (iAirlineClassList);

    return lCloneAirlineClassList;
  }

  // ////////////////////////////////////////////////////////////////////
  FlightPeriod& CmdCloneBomManager::
  cloneFlightPeriod (const FlightPeriod& iFlightPeriod) {

    FlightPeriod& lCloneFlightPeriod =
      FacCloneBom<FlightPeriod>::instance().clone (iFlightPeriod);

    // Check whether there are airline-class list objects
    const bool hasSegmentPeriodList =
      BomManager::hasList<SegmentPeriod> (iFlightPeriod);
    if (hasSegmentPeriodList == true) {
      // Browse the airline-class lists
      const SegmentPeriodList_T& lSegmentPeriodList =
        BomManager::getList<SegmentPeriod> (iFlightPeriod);
      for (SegmentPeriodList_T::const_iterator itSegmentPeriod =
             lSegmentPeriodList.begin();
           itSegmentPeriod != lSegmentPeriodList.end(); ++itSegmentPeriod) {
        const SegmentPeriod* lSegmentPeriod_ptr = *itSegmentPeriod;
        assert (lSegmentPeriod_ptr != NULL);

        // Clone the current airline-class list
        SegmentPeriod& lCloneSegmentPeriod =
          cloneSegmentPeriod (*lSegmentPeriod_ptr);
        FacBomManager::addToListAndMap (lCloneFlightPeriod,
                                        lCloneSegmentPeriod);
        FacBomManager::linkWithParent (lCloneFlightPeriod,
                                       lCloneSegmentPeriod);
      }
    }

    return lCloneFlightPeriod;
  }

  // ////////////////////////////////////////////////////////////////////
  SegmentPeriod& CmdCloneBomManager::
  cloneSegmentPeriod (const SegmentPeriod& iSegmentPeriod) {

```

```
    SegmentPeriod& lCloneSegmentPeriod =
      FacCloneBom<SegmentPeriod>::instance().clone (iSegmentPeriod);

    return lCloneSegmentPeriod;
  }

}

```

33.547 stdair/command/CmdCloneBomManager.hpp File Reference

```
#include <iosfwd>
#include <stdair/command/CmdAbstract.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/Inventory.hpp>
#include <stdair/bom/AirlineFeature.hpp>
#include <stdair/bom/OnDDate.hpp>
#include <stdair/bom/FlightDate.hpp>
#include <stdair/bom/LegDate.hpp>
#include <stdair/bom/LegCabin.hpp>
#include <stdair/bom/Bucket.hpp>
#include <stdair/bom/SegmentDate.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/FareFamily.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/bom/AirportPair.hpp>
#include <stdair/bom/PosChannel.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/bom/TimePeriod.hpp>
#include <stdair/bom/FareFeatures.hpp>
#include <stdair/bom/YieldFeatures.hpp>
#include <stdair/bom/AirlineClassList.hpp>
#include <stdair/bom/SegmentPeriod.hpp>
#include <stdair/bom/FlightPeriod.hpp>
```

Classes

- class stdair::CmdCloneBomManager

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.548 stdair/command/CmdCloneBomManager.hpp

```
00001 #ifndef __STDAIR_CMD_CMDCLONEBOMMANAGER_HPP
00002 #define __STDAIR_CMD_CMDCLONEBOMMANAGER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 // StdAir
00010 #include <stdair/command/CmdAbstract.hpp>
00011 // StdAir Bom
00012 #include <stdair/bom/BomRoot.hpp>
00013 #include <stdair/bom/Inventory.hpp>
00014 #include <stdair/bom/AirlineFeature.hpp>
00015 #include <stdair/bom/OnDDate.hpp>
00016 #include <stdair/bom/FlightDate.hpp>
00017 #include <stdair/bom/LegDate.hpp>
00018 #include <stdair/bom/LegCabin.hpp>
00019 #include <stdair/bom/Bucket.hpp>
00020 #include <stdair/bom/SegmentDate.hpp>
00021 #include <stdair/bom/SegmentCabin.hpp>
00022 #include <stdair/bom/FareFamily.hpp>
00023 #include <stdair/bom/BookingClass.hpp>
00024 #include <stdair/bom/AirportPair.hpp>
00025 #include <stdair/bom/PosChannel.hpp>
00026 #include <stdair/bom/DatePeriod.hpp>
00027 #include <stdair/bom/TimePeriod.hpp>
00028 #include <stdair/bom/FareFeatures.hpp>
00029 #include <stdair/bom/YieldFeatures.hpp>
00030 #include <stdair/bom/AirlineClassList.hpp>
00031 #include <stdair/bom/SegmentPeriod.hpp>
00032 #include <stdair/bom/FlightPeriod.hpp>
00033
00034 namespace stdair {
00035
00040   class CmdCloneBomManager : public CmdAbstract {
00041     //
00042     friend class STDAIR_Service;
00043   private:
00044
00051     static void cloneBomRoot (const BomRoot&, BomRoot&);
00052
00061     static Inventory& cloneInventory (const Inventory&, BomRoot&);
00062
00070     static AirlineFeature& cloneAirlineFeature (const AirlineFeature&);
00071
00079     static OnDDate& cloneOnDDate (const OnDDate&);
00080
00088     static FlightDate& cloneFlightDate (const FlightDate&);
00089
00097     static LegDate& cloneLegDate (const LegDate&);
00098
00106     static LegCabin& cloneLegCabin (const LegCabin&);
00107
00115     static Bucket& cloneBucket (const Bucket&);
00116
00124     static SegmentDate& cloneSegmentDate (const SegmentDate&);
00125
00133     static void linkBookingClassesWithSegment (SegmentDate&,
00134                                                SegmentCabin&);
00135
00143     static SegmentCabin& cloneSegmentCabin (const SegmentCabin&);
00144
00152     static FareFamily& cloneFareFamily (const FareFamily&);
00153
```

```
    static BookingClass& cloneBookingClass (const BookingClass&);

    static AirportPair& cloneAirportPair (const AirportPair&);

    static PosChannel& clonePosChannel (const PosChannel&);

    static DatePeriod& cloneDatePeriod (const DatePeriod&);

    static TimePeriod& cloneTimePeriod (const TimePeriod&);

    template <typename FEATURE_TYPE>
    static FEATURE_TYPE& cloneFeatures (const FEATURE_TYPE&);

    static AirlineClassList& cloneAirlineClassList (const AirlineClassList&);

    static FlightPeriod& cloneFlightPeriod (const FlightPeriod&);

    static SegmentPeriod& cloneSegmentPeriod (const SegmentPeriod&);

  };
}
#endif // ___STDAIR_CMD_CMDCLONEBOMMANAGER_HPP
```

33.549 stdair/command/DBManagerForAirlines.cpp File Reference

```
#include <cassert>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbaAirline.hpp>
#include <stdair/command/DBManagerForAirlines.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.550 stdair/command/DBManagerForAirlines.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 // SOCI
00007 #if defined(SOCI_HEADERS_BURIED)
00008 #include <soci/core/soci.h>
00009 #include <soci/backends/mysql/soci-mysql.h>
00010 #else // SOCI_HEADERS_BURIED
00011 #include <soci.h>
00012 #include <mysql/soci-mysql.h>
00013 #endif // SOCI_HEADERS_BURIED
00014 // StdAir
00015 #include <stdair/stdair_basic_types.hpp>
00016 #include <stdair/stdair_exceptions.hpp>
00017 #include <stdair/bom/AirlineStruct.hpp>
00018 #include <stdair/dbadaptor/DbaAirline.hpp>
00019 #include <stdair/command/DBManagerForAirlines.hpp>
00020 #include <stdair/service/Logger.hpp>
00021
00022 namespace stdair {
00023
00024   // ////////////////////////////////////////////////////////////////////
00025   void DBManagerForAirlines::
00026   prepareSelectStatement (DBSession_T& ioSociSession,
00027                           DBRequestStatement_T& ioSelectStatement,
00028                           AirlineStruct& ioAirline) {
00029
00030     try {
00031
00032       // Instanciate a SQL statement (no request is performed at that stage)
00038       ioSelectStatement = (ioSociSession.prepare
00039                            << "select iata_code, name "
00040                            << "from airlines ", soci::into (ioAirline));
00041
00042       // Execute the SQL query
00043       ioSelectStatement.execute();
00044
00045     } catch (std::exception const& lException) {
00046       throw SQLDatabaseException (lException.what());
00047     }
00048   }
00049
00050   // ////////////////////////////////////////////////////////////////////
00051   void DBManagerForAirlines::
00052   prepareSelectOnAirlineCodeStatement (DBSession_T& ioSociSession,
00053                                        DBRequestStatement_T& ioSelectStatement,
00054                                        const AirlineCode_T& iAirlineCode,
00055                                        AirlineStruct& ioAirline) {
00056
00057     try {
00058
00059       // Instanciate a SQL statement (no request is performed at that stage)
00066       ioSelectStatement = (ioSociSession.prepare
00067                            << "select iata_code, name "
00068                            << "from airlines "
00069                            << "where iata_code = :airline_code ",
00070                            soci::into (ioAirline), soci::use (iAirlineCode));
00071
00072       // Execute the SQL query
00073       ioSelectStatement.execute();
00074
00075     } catch (std::exception const& lException) {
00076       throw SQLDatabaseException (lException.what());
```

```
    }
  }

  // ////////////////////////////////////////////////////////////////////
  bool DBManagerForAirlines::
  iterateOnStatement (DBRequestStatement_T& ioStatement,
                      AirlineStruct& ioAirline) {
    bool hasStillData = false;

    try {

      // Retrieve the next row of Airline object
      hasStillData = ioStatement.fetch();

    } catch (std::exception const& lException) {
      throw SQLDatabaseException (lException.what());
    }

    return hasStillData;
  }

  // ////////////////////////////////////////////////////////////////////
  void DBManagerForAirlines::updateAirlineInDB (DBSession_T& ioSociSession,
                                                const AirlineStruct& iAirline) {
    try {
      // Begin a transaction on the database
      ioSociSession.begin();

      // Retrieve the airline code
      const std::string& lAirlineCode = iAirline.getAirlineCode();

      // Retrieve the airline name
      const std::string& lAirlineName = iAirline.getAirlineName();

      // Instanciate a SQL statement (no request is performed at that stage)
      DBRequestStatement_T lUpdateStatement =
        (ioSociSession.prepare
         << "update airlines "
         << "set name = :name "
         << "where iata_code = :iata_code",
         soci::use (lAirlineName), soci::use (lAirlineCode));

      // Execute the SQL query
      lUpdateStatement.execute (true);

      // Commit the transaction on the database
      ioSociSession.commit();

      // Debug
      // STDAIR_LOG_DEBUG ("[" << lAirlineCode << "] " << iAirline);

    } catch (std::exception const& lException) {
      throw SQLDatabaseException (lException.what());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  bool DBManagerForAirlines::retrieveAirline (DBSession_T& ioSociSession,
                                              const AirlineCode_T& iAirlineCode,
                                              AirlineStruct& ioAirline) {
    bool oHasRetrievedAirline = false;

    try {
      // Prepare the SQL request corresponding to the select statement
      DBRequestStatement_T lSelectStatement (ioSociSession);
      prepareSelectOnAirlineCodeStatement (ioSociSession, lSelectStatement,
                                           iAirlineCode, ioAirline);
```

```
00144
00145       // const bool shouldDoReset = true;
00146       bool hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00147       if (hasStillData == true) {
00148         oHasRetrievedAirline = true;
00149       }
00150
00151       // Sanity check
00152       // const bool shouldNotDoReset = false;
00153       hasStillData = iterateOnStatement (lSelectStatement, ioAirline);
00154
00155       // Debug
00156       // STDAIR_LOG_DEBUG ("[" << iDocID << "] " << ioAirline);
00157
00158     } catch (std::exception const& lException) {
00159       throw SQLDatabaseException (lException.what());
00160     }
00161
00162     return oHasRetrievedAirline;
00163   }
00164
00165 }
```

33.551 stdair/command/DBManagerForAirlines.hpp File Reference

```
#include <stdair/stdair_db.hpp>
#include <stdair/command/CmdAbstract.hpp>
```

Classes

- class stdair::DBManagerForAirlines

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.552 stdair/command/DBManagerForAirlines.hpp

```
#ifndef __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP
#define __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_db.hpp>
#include <stdair/command/CmdAbstract.hpp>

namespace stdair {

  // Forward declarations
  struct AirlineStruct;

  class DBManagerForAirlines : public CmdAbstract {
  public:
    static void updateAirlineInDB (DBSession_T&, const AirlineStruct&);

    static bool retrieveAirline (DBSession_T&, const AirlineCode_T&,
                                 AirlineStruct&);


  public:
    static void prepareSelectStatement (DBSession_T&, DBRequestStatement_T&,
                                        AirlineStruct&);

    static bool iterateOnStatement (DBRequestStatement_T&, AirlineStruct&);


  private:
    static void prepareSelectOnAirlineCodeStatement (DBSession_T&,
                                                     DBRequestStatement_T&,
                                                     const AirlineCode_T&,
                                                     AirlineStruct&);


  private:
    // ////////////////// Constructors and Destructors //////////////
    DBManagerForAirlines () {}
    DBManagerForAirlines (const DBManagerForAirlines&) {}
    ~DBManagerForAirlines () {}
  };

}
#endif // __TVLSIM_CMD_DBMANAGERFORAIRLINES_HPP
```

33.553 stdair/dbadaptor/DbaAbstract.cpp File Reference

```
#include <stdair/dbadaptor/DbaAbstract.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.554 stdair/dbadaptor/DbaAbstract.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/dbadaptor/DbaAbstract.hpp>

namespace stdair {

}
```

33.555 stdair/dbadaptor/DbaAbstract.hpp File Reference

```
#include <iosfwd>
```

Classes

- class stdair::DbaAbstract

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template<class charT , class traits >
  std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::DbaAbstract &iDba)
- template<class charT , class traits >
  std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::DbaAbstract &ioDba)

33.555.1 Function Documentation

33.555.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::DbaAbstract & *iDba*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 41 of file DbaAbstract.hpp.

33.555.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::DbaAbstract & *ioDba*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 69 of file DbaAbstract.hpp.

References stdair::DbaAbstract::fromStream().

33.556 stdair/dbadaptor/DbaAbstract.hpp

```
#ifndef __STDAIR_DBA_DBAABSTRACT_HPP
#define __STDAIR_DBA_DBAABSTRACT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>

namespace stdair {

  class DbaAbstract {
  public:

    virtual ~DbaAbstract() {}

    virtual void toStream (std::ostream& ioOut) const {}

    virtual void fromStream (std::istream& ioIn) {}

  protected:
    DbaAbstract() {}
  };
}

template <class charT, class traits>
inline
std::basic_ostream<charT, traits>&
operator<< (std::basic_ostream<charT, traits>& ioOut,
            const stdair::DbaAbstract& iDba) {
  std::basic_ostringstream<charT,traits> ostr;
  ostr.copyfmt (ioOut);
  ostr.width (0);

  // Fill string stream
  iDba.toStream (ostr);

  // Print string stream
  ioOut << ostr.str();

  return ioOut;
}

template <class charT, class traits>
inline
std::basic_istream<charT, traits>&
operator>> (std::basic_istream<charT, traits>& ioIn,
            stdair::DbaAbstract& ioDba) {
  // Fill Dba object with input stream
  ioDba.fromStream (ioIn);
  return ioIn;
}

#endif // __STDAIR_DBA_DBAABSTRACT_HPP
```

33.557 stdair/dbadaptor/DbaAirline.cpp File Reference

```
#include <exception>
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbaAirline.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace soci
- namespace stdair

  Handle on the StdAir library context.

33.558 stdair/dbadaptor/DbaAirline.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <exception>
#include <string>
// Stdair
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/bom/AirlineStruct.hpp>
#include <stdair/dbadaptor/DbaAirline.hpp>
#include <stdair/service/Logger.hpp>

namespace soci {

  // ////////////////////////////////////////////////////////////////////
  void type_conversion<stdair::AirlineStruct>::
  from_base (values const& iAirlineValues, indicator /* ind */,
             stdair::AirlineStruct& ioAirline) {
    /*
      iata_code, name
    */
    ioAirline.setAirlineCode (iAirlineValues.get<std::string> ("iata_code"));
    // The city code will be set to the default value (empty string)
    // when the column is null
    ioAirline.setAirlineName (iAirlineValues.get<std::string> ("name", ""));
  }

  // ////////////////////////////////////////////////////////////////////
  void type_conversion<stdair::AirlineStruct>::
  to_base (const stdair::AirlineStruct& iAirline, values& ioAirlineValues,
           indicator& ioIndicator) {
    const indicator lNameIndicator =
      iAirline.getAirlineName().empty() ? i_null : i_ok;
    ioAirlineValues.set ("iata_code", iAirline.getAirlineCode());
    ioAirlineValues.set ("name", iAirline.getAirlineName(), lNameIndicator);
    ioIndicator = i_ok;
  }

}

namespace stdair {

}
```

33.559 stdair/dbadaptor/DbaAirline.hpp File Reference

```
#include <soci/soci.h>
```

Classes

- struct soci::type_conversion< stdair::AirlineStruct >

Namespaces

- namespace stdair

  Handle on the StdAir library context.

- namespace soci

33.560 stdair/dbadaptor/DbaAirline.hpp

```
#ifndef __STDAIR_DBA_DBAAIRLINE_HPP
#define __STDAIR_DBA_DBAAIRLINE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// SOCI
#if defined(SOCI_HEADERS_BURIED)
#include <soci/core/soci.h>
#else // SOCI_HEADERS_BURIED
#include <soci/soci.h>
#endif // SOCI_HEADERS_BURIED

// Forward declarations
namespace stdair {
  struct AirlineStruct;
}

namespace soci {

  template <>
  struct type_conversion<stdair::AirlineStruct> {

    typedef values base_type;

    static void from_base (values const& iAirlineValues,
                           indicator /* ind */,
                           stdair::AirlineStruct& ioAirline);


    static void to_base (const stdair::AirlineStruct& iAirline,
                         values& ioAirlineValues,
                         indicator& ioIndicator);
  };
}
#endif // __STDAIR_DBA_DBAAIRLINE_HPP
```

33.561 stdair/factory/FacAbstract.cpp File Reference

```
#include <cassert>
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/factory/FacAbstract.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.562 stdair/factory/FacAbstract.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// STDAIR
#include <stdair/bom/BomAbstract.hpp>
#include <stdair/factory/FacAbstract.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FacAbstract::~FacAbstract() {
  }

}
```

33.563 stdair/factory/FacAbstract.hpp File Reference

Classes

- class stdair::FacAbstract

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.564 stdair/factory/FacAbstract.hpp

```
#ifndef __STDAIR_FAC_FACABSTRACT_HPP
#define __STDAIR_FAC_FACABSTRACT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////

namespace stdair {
  class FacAbstract {
  public:
    virtual ~FacAbstract();

  protected:
    FacAbstract() {}
  };
}
#endif // __STDAIR_FAC_FACABSTRACT_HPP
```

33.565 stdair/factory/FacBom.hpp File Reference

```
#include <cassert>
#include <string>
#include <list>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
```

Classes

- class stdair::FacBom< BOM >

  Base class for Factory layer.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.566 stdair/factory/FacBom.hpp

```
#ifndef __STDAIR_FAC_FACBOM_HPP
#define __STDAIR_FAC_FACBOM_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
#include <list>
// StdAir
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  template <typename BOM>
  class FacBom : public FacAbstract {

    typedef std::list<BOM*> BomPool_T;
    typedef typename BOM::Key_T Key_T;


  public:
    // ///////////// Business methods ///////////
    static FacBom& instance();

    BOM& create ();
    BOM& create (const Key_T&);
    BOM& create (const BOM&);

  protected:
    FacBom() {}

  public:
    ~FacBom() {
      clean();
    }

    void clean();


  private:
    // //////////////////// Attributes /////////////////
    static FacBom* _instance;

    BomPool_T _pool;
  };


  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> FacBom<BOM>* FacBom<BOM>::_instance = NULL;

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> FacBom<BOM>& FacBom<BOM>::instance () {
    if (_instance == NULL) {
      _instance = new FacBom ();
      assert (_instance != NULL);

      FacSupervisor::instance().registerPersistentBomFactory (_instance);
    }
    return *_instance;
  }

```

```
00094   // ////////////////////////////////////////////////////////////////////
00095   template <typename BOM> void FacBom<BOM>::clean () {
00096     // Destroy all the objects
00097     for (typename BomPool_T::iterator itBom = _pool.begin();
00098          itBom != _pool.end(); ++itBom) {
00099       BOM* currentBom_ptr = *itBom;
00100       assert (currentBom_ptr != NULL);
00101       delete currentBom_ptr; currentBom_ptr = NULL;
00102     }
00103
00104     // Empty the pool.
00105     _pool.clear();
00106
00107     // Reset the static instance.
00108     _instance = NULL;
00109   }
00110
00111   // ////////////////////////////////////////////////////////////////////
00112   template <typename BOM> BOM& FacBom<BOM>::create () {
00113     Key_T lKey;
00114     return instance().create (lKey);
00115   }
00116
00117   // ////////////////////////////////////////////////////////////////////
00118   template <typename BOM> BOM& FacBom<BOM>::create (const Key_T& iKey) {
00119     BOM* oBom_ptr = new BOM (iKey);
00120     assert (oBom_ptr != NULL);
00121     _pool.push_back (oBom_ptr);
00122     return *oBom_ptr;
00123   }
00124
00125   // ////////////////////////////////////////////////////////////////////
00126   template <typename BOM> BOM& FacBom<BOM>::create (const BOM& iBom) {
00127     BOM* oBom_ptr = new BOM (iBom);
00128     assert (oBom_ptr != NULL);
00129     _pool.push_back (oBom_ptr);
00130     return *oBom_ptr;
00131   }
00132
00133 }
00134 #endif // __STDAIR_FAC_FACBOM_HPP
```

33.567 stdair/factory/FacBomManager.cpp File Reference

```
#include <cassert>
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.568 stdair/factory/FacBomManager.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/basic/BasConst_Inventory.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/SegmentCabin.hpp>
#include <stdair/bom/SimpleNestingStructure.hpp>
#include <stdair/bom/NestingNode.hpp>
#include <stdair/bom/BookingClass.hpp>
#include <stdair/factory/FacBomManager.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {
  // ////////////////////////////////////////////////////////////////////
  void FacBomManager::
  resetYieldBasedNestingStructure (const SegmentCabin& iSegmentCabin) {
    const SimpleNestingStructure& lYieldBasedNS =
      BomManager::getObject<SimpleNestingStructure> (iSegmentCabin,
YIELD_BASED_NESTING_STRUCTURE_CODE);

    // Browse the list of node and reset each one.
    const NestingNodeList_T& lNestingNodeList =
      BomManager::getList<NestingNode> (lYieldBasedNS);
    for (NestingNodeList_T::const_iterator itNode = lNestingNodeList.begin();
         itNode != lNestingNodeList.end(); ++itNode) {
      stdair::NestingNode* lNode_ptr = *itNode;
      assert (lNode_ptr != NULL);

      lNode_ptr->setYield (-1.0);

      // Clear the list of booking classes of the node
      const HolderMap_T& lHolderMap = lNode_ptr->getHolderMap();
      HolderMap_T::const_iterator itHolder = lHolderMap.find (&typeid (
BookingClass));

      if (itHolder == lHolderMap.end()) {
        const std::string lName (typeid (BookingClass).name());
        throw NonInitialisedContainerException("Cannot find the holder of type "
                                               + lName + " within: "
                                               + lNode_ptr->describeKey());
      }

      BomHolder<BookingClass>* lBomHolder_ptr =
        static_cast<BomHolder<BookingClass>*> (itHolder->second);
      assert (lBomHolder_ptr != NULL);

      BookingClassList_T& lBCList = lBomHolder_ptr->_bomList;
      lBCList.clear();
    }
  }

}
```

33.569 stdair/factory/FacBomManager.hpp File Reference

```
#include <iosfwd>
#include <list>
#include <map>
#include <boost/static_assert.hpp>
#include <boost/type_traits/is_same.hpp>
#include <stdair/bom/BomHolder.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/bom/SegmentDate.hpp>
```

Classes

- class stdair::FacBomManager

  Utility class for linking StdAir-based objects.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.570 stdair/factory/FacBomManager.hpp

```
00001 #ifndef __STDAIR_FAC_FACBOMMANAGER_HPP
00002 #define __STDAIR_FAC_FACBOMMANAGER_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009 #include <list>
00010 #include <map>
00011 // Boost
00012 #include <boost/static_assert.hpp>
00013 #include <boost/type_traits/is_same.hpp>
00014 // StdAir
00015 #include <stdair/bom/BomHolder.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/factory/FacAbstract.hpp>
00018 #include <stdair/factory/FacBom.hpp>
00019 // Stdair BOM Objects
00020 #include <stdair/bom/SegmentDate.hpp>
00021
00022
00023 namespace stdair {
00024   // Forward declarations.
00025   class SegmentCabin;
00026
00030   class FacBomManager : public FacAbstract {
00031   public:
00032     // ///////////// Business methods. ////////////
00041     template <typename OBJECT2, typename OBJECT1>
00042     static BomHolder<OBJECT2>* getBomHolderPtr (OBJECT1&);
00043
00053     template <typename OBJECT2, typename OBJECT1>
00054     static BomHolder<OBJECT2>& addBomHolder (OBJECT1&);
00055
00067     template <typename OBJECT1, typename OBJECT2>
00068     static void addToList (OBJECT1&, OBJECT2&);
00069
00082     template <typename OBJECT1, typename OBJECT2>
00083     static void addToMap (OBJECT1&, OBJECT2&, const MapKey_T&);
00084
00096     template <typename OBJECT1, typename OBJECT2>
00097     static void addToMap (OBJECT1&, OBJECT2&);
00098
00110     template <typename OBJECT1, typename OBJECT2>
00111     static void addToListAndMap (OBJECT1&, OBJECT2&);
00112
00125     template <typename OBJECT1, typename OBJECT2>
00126     static void addToListAndMap (OBJECT1&, OBJECT2&, const MapKey_T&);
00127
00134     template <typename PARENT, typename CHILD>
00135     static void linkWithParent (PARENT&, CHILD&);
00136
00147     template <typename OBJECT2, typename OBJECT1>
00148     static void cloneHolder (OBJECT1&, const OBJECT1&);
00149
00150
00151   private:
00164     template <typename OBJECT1, typename OBJECT2>
00165     static void addToList (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&);
00166
00180     template <typename OBJECT1, typename OBJECT2>
00181     static void addToMap (BomHolder<OBJECT2>&, OBJECT1&, OBJECT2&,
00182                           const MapKey_T&);
00183
```

```
    template <typename OBJECT2, typename OBJECT1>
    static BomHolder<OBJECT2>& getBomHolder (OBJECT1&);

  public:
    static void resetYieldBasedNestingStructure (const SegmentCabin&);

    static void setAirlineFeature (Inventory& iInventory,
                                   AirlineFeature& iAirlineFeature) {
      iInventory.setAirlineFeature (iAirlineFeature);
    }

    static void linkWithOperating (SegmentDate& iSegmentDate,
                                   SegmentDate& iOperatingSegmentDate) {
      iSegmentDate.linkWithOperating (iOperatingSegmentDate);
    }


  protected:
    FacBomManager() { }

  public:
    ~FacBomManager() { }
  };

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertion to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT2, typename OBJECT1>
  BomHolder<OBJECT2>& FacBomManager::addBomHolder (OBJECT1& ioObject1) {

    //
    // Compile time assertion: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                          || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));


    BomHolder<OBJECT2>* lBomHolder_ptr =
      &FacBom<BomHolder<OBJECT2> >::instance().create();

    const bool hasInsertBeenSuccessful =
      ioObject1._holderMap.insert (typename HolderMap_T::
                                   value_type (&typeid (OBJECT2),
                                               lBomHolder_ptr)).second;
    assert (hasInsertBeenSuccessful == true);

    return *lBomHolder_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT2, typename OBJECT1>
  BomHolder<OBJECT2>* FacBomManager::getBomHolderPtr (OBJECT1& ioObject1) {

    //
    // Compile time assertion: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
```

```
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    BomHolder<OBJECT2>* lBomHolder_ptr = NULL;

    // Find the corresponding BomHolder within the object1, if existing.
    HolderMap_T::const_iterator itHolder =
      ioObject1._holderMap.find (&typeid (OBJECT2));

    if (itHolder != ioObject1._holderMap.end()) {
      lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);
    }

    return lBomHolder_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Private method.
  template <typename OBJECT2, typename OBJECT1>
  BomHolder<OBJECT2>& FacBomManager::getBomHolder (OBJECT1& ioObject1) {

    //
    // Compile time assertion: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    BomHolder<OBJECT2>* lBomHolder_ptr = NULL;

    // Find the corresponding BomHolder within the object1. If it does
    // not exist, then create one.
    HolderMap_T::const_iterator itHolder =
      ioObject1._holderMap.find (&typeid (OBJECT2));

    if (itHolder == ioObject1._holderMap.end()) {
      lBomHolder_ptr = &addBomHolder<OBJECT2, OBJECT1> (ioObject1);

    } else {
      lBomHolder_ptr = static_cast<BomHolder<OBJECT2>*> (itHolder->second);
    }

    assert (lBomHolder_ptr != NULL);

    return *lBomHolder_ptr;
  }

  // ////////////////////////////////////////////////////////////////////
  // Private method.
  template <typename OBJECT1, typename OBJECT2>
  void FacBomManager::addToList (BomHolder<OBJECT2>& ioBomHolder,
                                 OBJECT1& ioObject1, OBJECT2& ioObject2) {

    //
    // Compile time assertion: this function must never be called with the
    // following list of couple types:
```

```
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    ioBomHolder._bomList.push_back (&ioObject2);
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // This method is specialized for the following couple types:
  // <SegmentDate, SegmentDate>
  template <typename OBJECT1, typename OBJECT2>
  void FacBomManager::addToList (OBJECT1& ioObject1, OBJECT2& ioObject2) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);

    addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
  }

  // ////////////////////////////////////////////////////////////////////
  // Private method.
  template <typename OBJECT1, typename OBJECT2>
  void FacBomManager::addToMap (BomHolder<OBJECT2>& ioBomHolder,
                                OBJECT1& ioObject1, OBJECT2& ioObject2,
                                const MapKey_T& iKey) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    const bool insertionSucceeded =
      ioBomHolder._bomMap.insert (typename std::map<const MapKey_T, OBJECT2*>::
                                  value_type (iKey, &ioObject2)).second;

    if (insertionSucceeded == false) {
      // Build a nice message, so that the error be fully explicit
      std::ostringstream oStr;
      oStr << "The given object ('" << iKey
           << "') can not be added to the map of '" << ioObject1.describeKey()
```

```
            << "' object. That map already contains: '";

      unsigned int idx = 0;
      for (typename std::map<const MapKey_T, OBJECT2*>::const_iterator iter =
             ioBomHolder._bomMap.begin();
            iter != ioBomHolder._bomMap.end(); ++iter, ++idx) {
        const OBJECT2* lCurrentObject_ptr = iter->second;
        assert (lCurrentObject_ptr != NULL);

        if (idx != 0) {
          oStr << "; ";
        }
        oStr << lCurrentObject_ptr->describeKey();
      }
      oStr << "'";

      STDAIR_LOG_ERROR (oStr.str());
      throw ObjectLinkingException (oStr.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT1, typename OBJECT2> void FacBomManager::
  addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2, const MapKey_T& iKey) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                          || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);

    addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
  }

  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  // ////////////////////////////////////////////////////////////////////
  template <typename OBJECT1, typename OBJECT2>
  void FacBomManager::addToMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                          || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                          || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    const MapKey_T& lKey = ioObject2.describeKey();
    addToMap (ioObject1, ioObject2, lKey);
  }
```

```

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT1, typename OBJECT2>
  void FacBomManager::addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2,
                                       const MapKey_T& iKey) {
    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    BomHolder<OBJECT2>& lBomHolder = getBomHolder<OBJECT2> (ioObject1);

    addToList<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2);
    addToMap<OBJECT1, OBJECT2> (lBomHolder, ioObject1, ioObject2, iKey);
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method.
  // Compile time assertation to check OBJECT1 and OBJECT2 types.
  template <typename OBJECT1, typename OBJECT2> void FacBomManager::
  addToListAndMap (OBJECT1& ioObject1, OBJECT2& ioObject2) {

    //
    // Compile time assertation: this function must never be called with the
    // following list of couple types:
    // <SegmentDate, SegmentDate>
    // <AirlineFeature, Inventory>
    //
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, SegmentDate>::value == false
                         || boost::is_same<OBJECT2, SegmentDate>::value == false
    ));
    BOOST_STATIC_ASSERT ((boost::is_same<OBJECT1, Inventory>::value == false
                         || boost::is_same<OBJECT2, AirlineFeature>::value == fa
    lse));

    const MapKey_T& lKey = ioObject2.describeKey();
    addToListAndMap<OBJECT1, OBJECT2> (ioObject1, ioObject2, lKey);
  }

  // Public business method valid for all PARENT and CHILD types.
  // (No compile time assertation to check PARENT and CHILD types.)
  // ////////////////////////////////////////////////////////////////////
  template <typename PARENT, typename CHILD> void FacBomManager::
  linkWithParent (PARENT& ioParent,  CHILD& ioChild) {
    ioChild._parent = &ioParent;
  }

  // ////////////////////////////////////////////////////////////////////
  // Public business method valid for all PARENT and CHILD types.
  // (No compile time assertation to check PARENT and CHILD types.)
  template <typename OBJECT2, typename OBJECT1> void FacBomManager::
  cloneHolder (OBJECT1& ioDest, const OBJECT1& iOri) {

    const BomHolder<OBJECT2>& lOriginHolder =
      BomManager::getBomHolder<OBJECT2> (iOri);

    BomHolder<OBJECT2>& lDestHolder = getBomHolder<OBJECT2> (ioDest);
```

```
    lDestHolder._bomList = lOriginHolder._bomList;
    lDestHolder._bomMap = lOriginHolder._bomMap;
  }

  // ////////////////////////////////////////////////////////////////////
  //
  // Specialization of the template method addToList above for the types
  // <SegmentDate, SegmentDate>.
  // Add an element to the marketing segment date list of a segment date.
  //
  // ////////////////////////////////////////////////////////////////////

  template<>
  inline void FacBomManager::addToList <SegmentDate,SegmentDate>
  (SegmentDate& ioSegmentDate,
   SegmentDate& ioMarketingSegmentDate) {

    ioSegmentDate._marketingSegmentDateList.push_back(&ioMarketingSegmentDate);
  }

  // ////////////////////////////////////////////////////////////////////
  //
  // TODO:
  // This specialization is needed for all the objects in the current
  // BOM tree.
  // (An inventory is the parent of flight dates, a flight date is the
  // parent of segment dates and leg dates, ...)
  //
  // ////////////////////////////////////////////////////////////////////


}

// ////////////////////////////////////////////////////////////////////

#endif // __STDAIR_FAC_FACBOMMANAGER_HPP
```

33.571 stdair/factory/FacCloneBom.hpp File Reference

```
#include <cassert>
#include <string>
#include <list>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
```

Classes

- class stdair::FacCloneBom< BOM >

  Base class for Factory layer.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.572 stdair/factory/FacCloneBom.hpp

```
#ifndef __STDAIR_FAC_FACCLONEBOM_HPP
#define __STDAIR_FAC_FACCLONEBOM_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
#include <list>
// StdAir
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  template <typename BOM>
  class FacCloneBom : public FacAbstract {

    typedef std::list<BOM*> BomPool_T;
    typedef typename BOM::Key_T Key_T;


  public:
    // ////////////// Business methods ////////////
    static FacCloneBom& instance();

    BOM& clone (const BOM&);

  protected:
    FacCloneBom() {}

  public:
    ~FacCloneBom() {
      clean();
    }

    void clean();


  private:
    // //////////////////// Attributes /////////////////
    static FacCloneBom* _instance;

    BomPool_T _pool;
  };


  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> FacCloneBom<BOM>* FacCloneBom<BOM>::_instance = NULL;

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> FacCloneBom<BOM>& FacCloneBom<BOM>::instance () {
    if (_instance == NULL) {
      _instance = new FacCloneBom ();
      assert (_instance != NULL);

      FacSupervisor::instance().registerCloneBomFactory (_instance);
    }
    return *_instance;
  }

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> void FacCloneBom<BOM>::clean () {
```

```
    // Destroy all the objects
    for (typename BomPool_T::iterator itBom = _pool.begin();
         itBom != _pool.end(); ++itBom) {
      BOM* currentBom_ptr = *itBom;
      assert (currentBom_ptr != NULL);
      delete currentBom_ptr; currentBom_ptr = NULL;
    }

    // Empty the pool.
    _pool.clear();

    // Reset the static instance.
    _instance = NULL;
  }

  // ////////////////////////////////////////////////////////////////////
  template <typename BOM> BOM& FacCloneBom<BOM>::clone (const BOM& iBom) {
    BOM* oBom_ptr = new BOM (iBom);
    assert (oBom_ptr != NULL);
    _pool.push_back (oBom_ptr);
    return *oBom_ptr;
  }

}
#endif // __STDAIR_FAC_FACCLONEBOM_HPP
```

33.573 stdair/service/DBSessionManager.cpp File Reference

```
#include <cassert>
#include <string>
#include <sstream>
#include <soci.h>
#include <mysql/soci-mysql.h>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.574 stdair/service/DBSessionManager.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <string>
#include <sstream>
// SOCI
#if defined(SOCI_HEADERS_BURIED)
#include <soci/core/soci.h>
#include <soci/backends/mysql/soci-mysql.h>
#else // SOCI_HEADERS_BURIED
#include <soci.h>
#include <mysql/soci-mysql.h>
#endif // SOCI_HEADERS_BURIED
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  DBSessionManager::DBSessionManager () : _dbSession (NULL) {
  }

  // ////////////////////////////////////////////////////////////////////
  DBSessionManager::DBSessionManager (const DBSessionManager&)
    : _dbSession (NULL) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  DBSessionManager::~DBSessionManager () {
    // std::cout << "In DBSessionManager destructor" << std::endl;
    dbFinalise();
  }

  // ////////////////////////////////////////////////////////////////////
  void DBSessionManager::dbInit (const BasDBParams& iDBParams) {

    // Database parameters
    std::ostringstream oStr;
    oStr << "db=" << iDBParams.getDBName() << " user=" << iDBParams.getUser()
         << " password=" << iDBParams.getPassword()
         << " port=" << iDBParams.getPort() << " host=" << iDBParams.getHost();
    const std::string lDBSessionConnectionString (oStr.str());

    // Instanciate the database session: nothing else is performed at that stage
    _dbSession = new DBSession_T;

    try {
      // Open the connection to the database
      _dbSession->open (soci::mysql, lDBSessionConnectionString);

    } catch (std::exception const& lException) {
      std::ostringstream oMessage;
      oMessage <<"Error while opening a connection to database: "
               << lException.what() << std::endl
               << "Database parameters used:"
               << " db=" << iDBParams.getDBName()
               << " user=" << iDBParams.getUser()
               << " port=" << iDBParams.getPort()
               << " host=" << iDBParams.getHost();
```

```
      throw SQLDatabaseConnectionImpossibleException (oMessage.str());
    }
  }

  // ////////////////////////////////////////////////////////////////////
  void DBSessionManager::dbFinalise () {
    delete _dbSession; _dbSession = NULL;
  }

  // ////////////////////////////////////////////////////////////////////
  void DBSessionManager::init (const BasDBParams& iDBParams) {
    DBSessionManager& lInstance = instance();
    lInstance.dbInit (iDBParams);
  }

  // ////////////////////////////////////////////////////////////////////
  DBSessionManager& DBSessionManager::instance() {
    static DBSessionManager _instance;
    return _instance;
  }

  // ////////////////////////////////////////////////////////////////////
  void DBSessionManager::clean() {
  }

  // ////////////////////////////////////////////////////////////////////
  DBSession_T& DBSessionManager::getDBSession() const {
    if (_dbSession == NULL) {
      throw NonInitialisedDBSessionManagerException ("");
    }
    assert (_dbSession != NULL);
    return *_dbSession;
  }

}
```

33.575 stdair/service/DBSessionManager.hpp File Reference

```
#include <stdair/stdair_db.hpp>
```

Classes

- class stdair::DBSessionManager

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.576 stdair/service/DBSessionManager.hpp

```
#ifndef __STDAIR_SVC_DBSESSIONMANAGER_HPP
#define __STDAIR_SVC_DBSESSIONMANAGER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_db.hpp>

namespace stdair {

  // Forward declarations
  struct BasDBParams;

  class DBSessionManager {
    // Friend classes
    friend class FacSupervisor;
    friend class STDAIR_Service;

  public:
    static DBSessionManager& instance();

    DBSession_T& getDBSession() const;


  private:
    DBSessionManager ();
    DBSessionManager (const DBSessionManager&);
    ~DBSessionManager ();

    void dbInit (const BasDBParams&);

    void dbFinalise ();


  private:
    static void init (const BasDBParams&);

    static void clean();


  private:
    DBSession_T* _dbSession;
  };

}
#endif // __STDAIR_SVC_DBSESSIONMANAGER_HPP
```

33.577 stdair/service/FacServiceAbstract.cpp File Reference

```
#include <cassert>
#include <stdair/service/ServiceAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.578 stdair/service/FacServiceAbstract.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/service/ServiceAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  FacServiceAbstract::~FacServiceAbstract() {
    clean();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacServiceAbstract::clean() {
    for (ServicePool_T::iterator itService = _pool.begin();
         itService != _pool.end(); itService++) {
      ServiceAbstract* currentService_ptr = *itService;
      assert (currentService_ptr != NULL);

      delete (currentService_ptr); currentService_ptr = NULL;
    }

    // Empty the pool of Service Factories
    _pool.clear();
  }

}
```

33.579 stdair/service/FacServiceAbstract.hpp File Reference

```
#include <vector>
```

Classes

- class stdair::FacServiceAbstract

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.580 stdair/service/FacServiceAbstract.hpp

```
#ifndef __STDAIR_SVC_FACSERVICEABSTRACT_HPP
#define __STDAIR_SVC_FACSERVICEABSTRACT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <vector>

namespace stdair {

  // Forward declarations
  class ServiceAbstract;

  class FacServiceAbstract {
  public:

    typedef std::vector<ServiceAbstract*> ServicePool_T;

    virtual ~FacServiceAbstract();

    void clean();

  protected:
    FacServiceAbstract() {}

    ServicePool_T _pool;
  };

}
#endif // __STDAIR_SVC_FACSERVICEABSTRACT_HPP
```

33.581 stdair/service/FacSTDAIRServiceContext.cpp File Reference

```
#include <cassert>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.582 stdair/service/FacSTDAIRServiceContext.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>

namespace stdair {

  FacSTDAIRServiceContext* FacSTDAIRServiceContext::_instance = NULL;

  // ////////////////////////////////////////////////////////////////////
  FacSTDAIRServiceContext::~FacSTDAIRServiceContext() {
    _instance = NULL;
  }

  // ////////////////////////////////////////////////////////////////////
  FacSTDAIRServiceContext& FacSTDAIRServiceContext::instance() {

    if (_instance == NULL) {
      _instance = new FacSTDAIRServiceContext();
      assert (_instance != NULL);

      FacSupervisor::instance().registerServiceFactory (_instance);
    }
    return *_instance;
  }

  // ////////////////////////////////////////////////////////////////////
  STDAIR_ServiceContext& FacSTDAIRServiceContext::create() {
    STDAIR_ServiceContext* aServiceContext_ptr = NULL;

    aServiceContext_ptr = new STDAIR_ServiceContext ();
    assert (aServiceContext_ptr != NULL);

    // The new object is added to the Bom pool
    _pool.push_back (aServiceContext_ptr);

    return *aServiceContext_ptr;
  }

}
```

33.583 stdair/service/FacSTDAIRServiceContext.hpp File Reference

```
#include <stdair/service/FacServiceAbstract.hpp>
```

Classes

- class stdair::FacSTDAIRServiceContext

  Factory for Bucket.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.584 stdair/service/FacSTDAIRServiceContext.hpp

```
#ifndef __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
#define __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/service/FacServiceAbstract.hpp>

namespace stdair {

  class STDAIR_ServiceContext;

  class FacSTDAIRServiceContext : public FacServiceAbstract {
  public:

    static FacSTDAIRServiceContext& instance();

    ~FacSTDAIRServiceContext();

    STDAIR_ServiceContext& create();


  protected:
    FacSTDAIRServiceContext() {}

  private:
    static FacSTDAIRServiceContext* _instance;
  };

}
#endif // __STDAIR_SVC_FACSTDAIRSERVICECONTEXT_HPP
```

33.585 stdair/service/FacSupervisor.cpp File Reference

```
#include <cassert>
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.586 stdair/service/FacSupervisor.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/factory/FacAbstract.hpp>
#include <stdair/service/FacServiceAbstract.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>

namespace stdair {

  FacSupervisor* FacSupervisor::_instance = NULL;

  // ////////////////////////////////////////////////////////////////////
  FacSupervisor& FacSupervisor::instance() {
    if (_instance == NULL) {
      _instance = new FacSupervisor();
    }

    return *_instance;
  }

  // ////////////////////////////////////////////////////////////////////
  FacSupervisor::~FacSupervisor() {
    cleanPersistentBomLayer();
    cleanCloneBomLayer();
    cleanServiceLayer();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::registerPersistentBomFactory (FacAbstract* ioFac_ptr) {
    _persistentBomPool.push_back (ioFac_ptr);
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::registerCloneBomFactory (FacAbstract* ioFac_ptr) {
    _cloneBomPool.push_back (ioFac_ptr);
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::registerServiceFactory (FacServiceAbstract* ioFac_ptr) {
    _svcPool.push_back (ioFac_ptr);
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanPersistentBomLayer() {
    for (PersistentBomFactoryPool_T::const_iterator itFactory = _persistentBomPool.begin();
         itFactory != _persistentBomPool.end(); itFactory++) {
      const FacAbstract* currentFactory_ptr = *itFactory;
      assert (currentFactory_ptr != NULL);

      delete (currentFactory_ptr); currentFactory_ptr = NULL;
    }
    // Empty the pool of BOM factories
    _persistentBomPool.clear();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanCloneBomLayer() {
    for (CloneBomFactoryPool_T::const_iterator itFactory = _cloneBomPool.begin();
```

```
         itFactory != _cloneBomPool.end(); itFactory++) {
      const FacAbstract* currentFactory_ptr = *itFactory;
      assert (currentFactory_ptr != NULL);

      delete (currentFactory_ptr); currentFactory_ptr = NULL;
    }

    // Empty the pool of BOM factories
    _cloneBomPool.clear();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanServiceLayer() {
    for (ServiceFactoryPool_T::const_iterator itFactory = _svcPool.begin();
         itFactory != _svcPool.end(); itFactory++) {
      const FacServiceAbstract* currentFactory_ptr = *itFactory;
      assert (currentFactory_ptr != NULL);

      delete (currentFactory_ptr); currentFactory_ptr = NULL;
    }

    // Empty the pool of Service Factories
    _svcPool.clear();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanLoggerService() {
    // Clean the static instance of the log service
    Logger::clean();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanDBSessionManager() {
    // Clean the static instance of the database service
    DBSessionManager::clean();
  }

  // ////////////////////////////////////////////////////////////////////
  void FacSupervisor::cleanAll() {

    // Clean the static instance of the database session manager
    cleanDBSessionManager();

    // Clean the static instance of the log service
    cleanLoggerService();

    // Clean the static instance of the FacSupervisor.
    // This in turn will invoke the destructor (~FacSupervisor() method)
    // of the static instance, thus cleaning both the BOM and service layers.
    delete _instance; _instance = NULL;
  }

}
```

33.587 stdair/service/FacSupervisor.hpp File Reference

```
#include <iosfwd>
#include <list>
```

Classes

- class stdair::FacSupervisor

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.588 stdair/service/FacSupervisor.hpp

```
#ifndef __STDAIR_SVC_FACSUPERVISOR_HPP
#define __STDAIR_SVC_FACSUPERVISOR_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iosfwd>
#include <list>

namespace stdair {

  class FacAbstract;
  class FacServiceAbstract;

  class FacSupervisor {
  public:
    typedef std::list<FacAbstract*> PersistentBomFactoryPool_T;
    typedef std::list<FacAbstract*> CloneBomFactoryPool_T;
    typedef std::list<FacServiceAbstract*> ServiceFactoryPool_T;

    static FacSupervisor& instance();

    void registerPersistentBomFactory (FacAbstract*);

    void registerCloneBomFactory (FacAbstract*);

    void registerServiceFactory (FacServiceAbstract*);

    void cleanPersistentBomLayer();

    void cleanCloneBomLayer();

    void cleanServiceLayer();

    static void cleanLoggerService();

    static void cleanDBSessionManager();

    static void cleanAll();

    ~FacSupervisor();


  protected:
    FacSupervisor() {}
    FacSupervisor (const FacSupervisor&) {}

  private:
    static FacSupervisor* _instance;

    PersistentBomFactoryPool_T _persistentBomPool;

    CloneBomFactoryPool_T _cloneBomPool;

    ServiceFactoryPool_T _svcPool;

  };
}
#endif // __STDAIR_SVC_FACSUPERVISOR_HPP
```

33.589 stdair/service/Logger.cpp File Reference

```
#include <iostream>
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.590 stdair/service/Logger.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <iostream>
// StdAir Logger
#include <stdair/stdair_exceptions.hpp>
#include <stdair/service/Logger.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  Logger::Logger()
    : _level (LOG::DEBUG), _logStream (&std::cout),
      _hasBeenInitialised (false) {
  }

  // ////////////////////////////////////////////////////////////////////
  Logger::Logger (const Logger&)
    : _level (LOG::DEBUG), _logStream (&std::cout),
      _hasBeenInitialised (false) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  Logger::~Logger() {
    // std::cout << "In Logger destructor" << std::endl;
  }

  // ////////////////////////////////////////////////////////////////////
  void Logger::init (const BasLogParams& iLogParams) {

    //
    Logger& lInstance = instance();
    const bool hasBeenInitialised = lInstance.getStatus();
    if (hasBeenInitialised == true
        && iLogParams.getForcedInitialisationFlag() == false) {
      STDAIR_LOG_ERROR ("Error: the log stream has already been initialised");
      assert (false);
    }

    lInstance.setLevel (iLogParams._logLevel);
    lInstance.setStream (iLogParams._logStream);
    lInstance.setStatus (true);
  }

  // ////////////////////////////////////////////////////////////////////
  Logger& Logger::instance() {
    static Logger _instance;
    return _instance;
  }

  // ////////////////////////////////////////////////////////////////////
  BasLogParams Logger::getLogParams() {
    std::ostream* oStream_ptr = instance()._logStream;
    assert (oStream_ptr != NULL);
    return BasLogParams (instance()._level, *oStream_ptr);
  }

  // ////////////////////////////////////////////////////////////////////
  void Logger::clean() {
    Logger& lInstance = instance();
    lInstance.setStatus (false);
  }

```

```
00066 }
```

33.591 stdair/service/Logger.hpp File Reference

```
#include <cassert>
#include <sstream>
#include <string>
#include <stdair/stdair_log.hpp>
#include <stdair/basic/BasLogParams.hpp>
```

Classes

- class stdair::Logger

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Defines

- #define STDAIR_LOG_CORE(iLevel, iToBeLogged)
- #define STDAIR_LOG_CRITICAL(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)
- #define STDAIR_LOG_ERROR(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)
- #define STDAIR_LOG_NOTIFICATION(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)
- #define STDAIR_LOG_WARNING(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)
- #define STDAIR_LOG_DEBUG(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)
- #define STDAIR_LOG_VERBOSE(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)

33.591.1 Define Documentation

33.591.1.1 #define STDAIR_LOG_CORE(iLevel, iToBeLogged)

Value:

```
{ std::ostringstream ostr; ostr << iToBeLogged; \
    stdair::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }
```

Definition at line 16 of file Logger.hpp.

33.591.1.2 #define STDAIR_LOG_CRITICAL(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)

Definition at line 20 of file Logger.hpp.

33.591.1.3 #define STDAIR_LOG_ERROR(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)

Definition at line 23 of file Logger.hpp.

Referenced by stdair::ParsedKey::getBoardingTime(), stdair::ParsedKey::getFlightDateKey(), stdair::ParsedKey::getInventoryKey(), stdair::ParsedKey::getLegKey(), stdair::BomManager::getObject(), stdair::ParsedKey::getSegmentKey(), and stdair::ConfigHolderStruct::updateAirlineFeatures().

33.591.1.4 #define STDAIR_LOG_NOTIFICATION(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)

Definition at line 26 of file Logger.hpp.

33.591.1.5 #define STDAIR_LOG_WARNING(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)

Definition at line 29 of file Logger.hpp.

33.591.1.6 #define STDAIR_LOG_DEBUG(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)

Definition at line 32 of file Logger.hpp.

Referenced by stdair::FRAT5CurveHolderStruct::addCurve(), stdair::FFDisutilityCurveHolderStruct::addCurve(), stdair::ParsedKey::getBoardingTime(), stdair::FFDisutilityCurveHolderStruct::getFFDisutilityCurve(), stdair::ParsedKey::getFlightDateKey(), stdair::FRAT5CurveHolderStruct::getFRAT5Curve(), stdair::ParsedKey::getInventoryKey(), stdair::ParsedKey::getLegKey(), stdair::ParsedKey::getSegmentKey(), stdair::BomINIImport::importINIConfig(), stdair::TimePeriod::isDepartureTimeValid(), and stdair::BomRetriever::retrieveSegmentDateFromLongKey().

33.591.1.7 #define STDAIR_LOG_VERBOSE(iToBeLogged) STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)

Definition at line 35 of file Logger.hpp.

33.592 stdair/service/Logger.hpp

```
#ifndef __STDAIR_SVC_LOGGER_HPP
#define __STDAIR_SVC_LOGGER_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
#include <string>
// StdAir
#include <stdair/stdair_log.hpp>
#include <stdair/basic/BasLogParams.hpp>

// /////////////// LOG MACROS /////////////////
#define STDAIR_LOG_CORE(iLevel, iToBeLogged) \
  { std::ostringstream ostr; ostr << iToBeLogged; \
    stdair::Logger::instance().log (iLevel, __LINE__, __FILE__, ostr.str()); }

#define STDAIR_LOG_CRITICAL(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::CRITICAL, iToBeLogged)

#define STDAIR_LOG_ERROR(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::ERROR, iToBeLogged)

#define STDAIR_LOG_NOTIFICATION(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::NOTIFICATION, iToBeLogged)

#define STDAIR_LOG_WARNING(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::WARNING, iToBeLogged)

#define STDAIR_LOG_DEBUG(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::DEBUG, iToBeLogged)

#define STDAIR_LOG_VERBOSE(iToBeLogged) \
  STDAIR_LOG_CORE (stdair::LOG::VERBOSE, iToBeLogged)
// /////////// (END OF) LOG MACROS /////////////


namespace stdair {

  class Logger {
    friend class FacSupervisor;
    friend class STDAIR_Service;

  public:

    template <typename T>
    void log (const LOG::EN_LogLevel iLevel, const int iLineNumber,
              const std::string& iFileName, const T& iToBeLogged) {
      assert (_logStream != NULL);
      if (iLevel <= _level) {
        *_logStream << "[" << LOG::_logLevels[iLevel] << "]" << iFileName << ":"
                    << iLineNumber << ": " << iToBeLogged << std::endl;
      }
    }

    static Logger& instance();


  private:
    // ////////////////// Initialisation and finalisation ///////////////
    bool getStatus() const {
      return _hasBeenInitialised;
    }
```

```
00082
00086     void setLevel (const LOG::EN_LogLevel& iLevel) {
00087       _level = iLevel;
00088     }
00089
00093     void setStream (std::ostream& ioStream) {
00094       _logStream = &ioStream;
00095     }
00096
00100     void setStatus (const bool iStatus) {
00101       _hasBeenInitialised = iStatus;
00102     }
00103
00108     Logger();
00112     Logger (const Logger&);
00116     ~Logger();
00117
00123     static void init (const BasLogParams&);
00124
00128     static BasLogParams getLogParams();
00129
00133     static void clean();
00134
00135
00136   private:
00137     // ////////////////// Attributes ///////////////
00141     LOG::EN_LogLevel _level;
00142
00146     std::ostream* _logStream;
00147
00151     bool _hasBeenInitialised;
00152   };
00153
00154 }
00155 #endif // __STDAIR_SVC_LOGGER_HPP
00156
```

33.593 stdair/service/ServiceAbstract.cpp File Reference

```
#include <stdair/service/ServiceAbstract.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.594 stdair/service/ServiceAbstract.cpp

```
// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/service/ServiceAbstract.hpp>

namespace stdair {

}
```

33.595 stdair/service/ServiceAbstract.hpp File Reference

```
#include <iosfwd>
```

Classes

- class stdair::ServiceAbstract

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Functions

- template<class charT , class traits >
  std::basic_ostream< charT, traits > & operator<< (std::basic_ostream< charT, traits > &ioOut, const stdair::ServiceAbstract &iService)
- template<class charT , class traits >
  std::basic_istream< charT, traits > & operator>> (std::basic_istream< charT, traits > &ioIn, stdair::ServiceAbstract &ioService)

33.595.1 Function Documentation

33.595.1.1 template<class charT , class traits > std::basic_ostream<charT, traits>& operator<< (std::basic_ostream< charT, traits > & *ioOut*, const stdair::ServiceAbstract & *iService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (p653) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 58 of file ServiceAbstract.hpp.

33.595.1.2 template<class charT , class traits > std::basic_istream<charT, traits>& operator>> (std::basic_istream< charT, traits > & *ioIn*, stdair::ServiceAbstract & *ioService*) [inline]

Piece of code given by Nicolai M. Josuttis, Section 13.12.1 "Implementing Output Operators" (pp655-657) of his book "The C++ Standard Library: A Tutorial and Reference", published by Addison-Wesley.

Definition at line 86 of file ServiceAbstract.hpp.

References stdair::ServiceAbstract::fromStream().

33.596 stdair/service/ServiceAbstract.hpp

```
00001 #ifndef __STDAIR_SVC_SERVICEABSTRACT_HPP
00002 #define __STDAIR_SVC_SERVICEABSTRACT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <iosfwd>
00009
00010 namespace stdair {
00011
00015   class ServiceAbstract {
00016   public:
00017
00021     virtual ~ServiceAbstract() {}
00022
00028     virtual void toStream (std::ostream& ioOut) const {}
00029
00035     virtual void fromStream (std::istream& ioIn) {}
00036
00040     // virtual const std::string describe() const = 0;
00041
00042   protected:
00046     ServiceAbstract() {}
00047   };
00048 }
00049
00055 template <class charT, class traits>
00056 inline
00057 std::basic_ostream<charT, traits>&
00058 operator<< (std::basic_ostream<charT, traits>& ioOut,
00059             const stdair::ServiceAbstract& iService) {
00065   std::basic_ostringstream<charT,traits> ostr;
00066   ostr.copyfmt (ioOut);
00067   ostr.width (0);
00068
00069   // Fill string stream
00070   iService.toStream (ostr);
00071
00072   // Print string stream
00073   ioOut << ostr.str();
00074
00075   return ioOut;
00076 }
00077
00083 template <class charT, class traits>
00084 inline
00085 std::basic_istream<charT, traits>&
00086 operator>> (std::basic_istream<charT, traits>& ioIn,
00087             stdair::ServiceAbstract& ioService) {
00088   // Fill Service object with input stream
00089   ioService.fromStream (ioIn);
00090   return ioIn;
00091 }
00092
00093 #endif // __STDAIR_SVC_SERVICEABSTRACT_HPP
```

33.597 stdair/service/STDAIR_Service.cpp File Reference

```
#include <cassert>
#include <sstream>
#include <stdair/stdair_types.hpp>
#include <stdair/stdair_json.hpp>
#include <stdair/basic/BasChronometer.hpp>
#include <stdair/bom/BomManager.hpp>
#include <stdair/bom/BomRetriever.hpp>
#include <stdair/bom/BomJSONExport.hpp>
#include <stdair/bom/BomJSONImport.hpp>
#include <stdair/bom/BomINIImport.hpp>
#include <stdair/bom/BomDisplay.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/bom/EventStruct.hpp>
#include <stdair/bom/BookingRequestStruct.hpp>
#include <stdair/bom/DatePeriod.hpp>
#include <stdair/command/CmdBomManager.hpp>
#include <stdair/command/CmdCloneBomManager.hpp>
#include <stdair/service/FacSupervisor.hpp>
#include <stdair/service/FacSTDAIRServiceContext.hpp>
#include <stdair/service/Logger.hpp>
#include <stdair/service/DBSessionManager.hpp>
#include <stdair/STDAIR_Service.hpp>
```

Namespaces

- namespace bpt
- namespace stdair

  Handle on the StdAir library context.

33.598 stdair/service/STDAIR_Service.cpp

```
00001 // ////////////////////////////////////////////////////////////////////
00002 // Import section
00003 // ////////////////////////////////////////////////////////////////////
00004 // STL
00005 #include <cassert>
00006 #include <sstream>
00007 #if BOOST_VERSION >= 104100
00008 // Boost Property Tree
00009 #include <boost/property_tree/ptree.hpp>
00010 #include <boost/property_tree/json_parser.hpp>
00011 #endif // BOOST_VERSION >= 104100
00012 // StdAir
00013 #include <stdair/stdair_types.hpp>
00014 #include <stdair/stdair_json.hpp>
00015 #include <stdair/basic/BasChronometer.hpp>
00016 #include <stdair/bom/BomManager.hpp>
00017 #include <stdair/bom/BomRetriever.hpp>
00018 #include <stdair/bom/BomJSONExport.hpp>
00019 #include <stdair/bom/BomJSONImport.hpp>
00020 #include <stdair/bom/BomINIImport.hpp>
00021 #include <stdair/bom/BomDisplay.hpp>
00022 #include <stdair/bom/BomRoot.hpp>
00023 #include <stdair/bom/EventStruct.hpp>
00024 #include <stdair/bom/BookingRequestStruct.hpp>
00025 #include <stdair/bom/DatePeriod.hpp>
00026 #include <stdair/command/CmdBomManager.hpp>
00027 #include <stdair/command/CmdCloneBomManager.hpp>
00028 #include <stdair/service/FacSupervisor.hpp>
00029 #include <stdair/service/FacSTDAIRServiceContext.hpp>
00030 #include <stdair/service/Logger.hpp>
00031 #include <stdair/service/DBSessionManager.hpp>
00032 #include <stdair/STDAIR_Service.hpp>
00033
00034 #if BOOST_VERSION >= 104100
00035 namespace bpt = boost::property_tree;
00036 #else // BOOST_VERSION >= 104100
00037 namespace bpt {
00038   typedef char ptree;
00039 }
00040 #endif // BOOST_VERSION >= 104100
00041
00042 namespace stdair {
00043
00044   // ////////////////////////////////////////////////////////////////////
00045   STDAIR_Service::STDAIR_Service() : _stdairServiceContext (NULL) {
00046
00047     // Initialise the service context
00048     initServiceContext();
00049
00050     // Initialise the (remaining of the) context
00051     init();
00052   }
00053
00054   // ////////////////////////////////////////////////////////////////////
00055   STDAIR_Service::STDAIR_Service (const STDAIR_Service& iService)
00056     : _stdairServiceContext (NULL) {
00057     assert (false);
00058   }
00059
00060   // ////////////////////////////////////////////////////////////////////
00061   STDAIR_Service::STDAIR_Service (const BasLogParams& iLogParams)
00062     : _stdairServiceContext (NULL) {
00063
00064     // Initialise the service context
00065     initServiceContext();
```

```

  // Set the log file
  logInit (iLogParams);

  // Initialise the (remaining of the) context
  init();
}

// ////////////////////////////////////////////////////////////////////
STDAIR_Service::STDAIR_Service (const BasLogParams& iLogParams,
                                const BasDBParams& iDBParams)
  : _stdairServiceContext (NULL) {

  // Initialise the service context
  initServiceContext();

  // Set the log file
  logInit (iLogParams);

  // Create a database session
  dbInit (iDBParams);

  // Initialise the (remaining of the) context
  init();
}

// ////////////////////////////////////////////////////////////////////
STDAIR_Service::~STDAIR_Service() {
  // Delete/Clean all the objects from memory
  finalise();
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::initServiceContext() {
  // Initialise the service context
  STDAIR_ServiceContext& lSTDAIR_ServiceContext =
    FacSTDAIRServiceContext::instance().create();

  // Store the stdair service context
  _stdairServiceContext = &lSTDAIR_ServiceContext;
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::logInit (const BasLogParams& iLogParams) {
  Logger::init (iLogParams);
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::dbInit (const BasDBParams& iDBParams) {
  DBSessionManager::init (iDBParams);

  // Store the database parameters into the StdAir service context
  assert (_stdairServiceContext != NULL);
  STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
  lSTDAIR_ServiceContext.setDBParams (iDBParams);
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::init() {
}

// ////////////////////////////////////////////////////////////////////
BomRoot& STDAIR_Service::getBomRoot() const {
  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
  const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
    *_stdairServiceContext;
```

```
    // Return the clone built-in Bom root
    return lSTDAIR_ServiceContext.getCloneBomRoot();
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot& STDAIR_Service::getPersistentBomRoot() const {
    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;
    // Return the persistent built-in Bom root
    return lSTDAIR_ServiceContext.getPersistentBomRoot();
  }

  // ////////////////////////////////////////////////////////////////////
  BasLogParams STDAIR_Service::getLogParams() const {
    return Logger::getLogParams();
  }

  // ////////////////////////////////////////////////////////////////////
  const BasDBParams& STDAIR_Service::getDBParams() const {
    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;
    return lSTDAIR_ServiceContext.getDBParams();
  }

  // ////////////////////////////////////////////////////////////////////
  const ServiceInitialisationType& STDAIR_Service::
  getServiceInitialisationType() const {
    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;
    return lSTDAIR_ServiceContext.getServiceInitialisationType();
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_Service::buildSampleBom() {
    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the BOM tree root
    BomRoot& lPersistentBomRoot = lSTDAIR_ServiceContext.getPersistentBomRoot();

    // Delegate the building process to the dedicated command
    CmdBomManager::buildSampleBom (lPersistentBomRoot);
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_Service::
  buildDummyInventory (const CabinCapacity_T& iCabinCapacity) {
    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the BOM tree root
    BomRoot& lPersistentBomRoot = lSTDAIR_ServiceContext.getPersistentBomRoot();

    // Delegate the building process to the dedicated command
    CmdBomManager::buildDummyInventory (lPersistentBomRoot, iCabinCapacity);
    CmdBomManager::buildCompleteDummyInventoryForFareFamilies (lPersistentBomRoot
);
```

```
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::
buildDummyLegSegmentAccesses (BomRoot& iBomRoot) {
  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);

  // Delegate the building process to the dedicated command
  CmdBomManager::buildDummyLegSegmentAccesses (iBomRoot);

}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::
buildSampleTravelSolutionForPricing (TravelSolutionList_T& ioTravelSolutionList
) {
  // Build a sample list of travel solution structures
  CmdBomManager::buildSampleTravelSolutionForPricing (ioTravelSolutionList);
}

// ////////////////////////////////////////////////////////////////////
void STDAIR_Service::
buildSampleTravelSolutions (TravelSolutionList_T& ioTravelSolutionList) {
  // Build a sample list of travel solution structures
  CmdBomManager::buildSampleTravelSolutions (ioTravelSolutionList);
}

// ////////////////////////////////////////////////////////////////////
BookingRequestStruct STDAIR_Service::
buildSampleBookingRequest (const bool isForCRS) {

  // Build a sample booking request structure
  if (isForCRS == true) {
    return CmdBomManager::buildSampleBookingRequestForCRS();
  }

  return CmdBomManager::buildSampleBookingRequest();
}


// ////////////////////////////////////////////////////////////////////
std::string STDAIR_Service::
jsonExportFlightDateList (const AirlineCode_T& iAirlineCode,
                          const FlightNumber_T& iFlightNumber) const {
  std::ostringstream oStr;

  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
  const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
    *_stdairServiceContext;

  // Retrieve the BOM tree root
  const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

  BomJSONExport::jsonExportFlightDateList (oStr, lCloneBomRoot,
                                           iAirlineCode, iFlightNumber);

  return oStr.str();
}

// ////////////////////////////////////////////////////////////////////
std::string STDAIR_Service::
jsonExportFlightDateObjects (const stdair::AirlineCode_T& iAirlineCode,
                             const stdair::FlightNumber_T& iFlightNumber,
                             const stdair::Date_T& iDepartureDate) const {
```

```
    std::ostringstream oStr;

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the BOM tree root
    const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

    // Retrieve the flight-date object corresponding to the key
    FlightDate* lFlightDate_ptr =
      BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot, iAirlineCode,
                                                  iFlightNumber,
                                                  iDepartureDate);

    // Dump the content of the whole BOM tree into the string
    if (lFlightDate_ptr != NULL) {
      BomJSONExport::jsonExportFlightDateObjects (oStr, *lFlightDate_ptr);

    } else {
#if BOOST_VERSION >= 104100
      //
      bpt::ptree lPropertyTree;

      // Build the appropriate message, so that the client may know that
      // no flight-date can be found for that given key.
      std::ostringstream oNoFlightDateStream;
      oNoFlightDateStream << "No flight-date found for the given key: '"
                          << iAirlineCode << iFlightNumber
                          << " - " << iDepartureDate << "'";
      const std::string oNoFlightDateString (oNoFlightDateStream.str());

      // Put in the property tree the fact that no flight-date has been found.
      // \note That is not (necessary) an error.
      lPropertyTree.put ("error", oNoFlightDateString.c_str());

      // Write the property tree into the JSON stream.
      write_json (oStr, lPropertyTree);
#endif // BOOST_VERSION >= 104100
    }

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::
  jsonExportEventObject (const EventStruct& iEventStruct) const {

    std::ostringstream oStr;

    const EventType::EN_EventType& lEventType =
      iEventStruct.getEventType();

    switch (lEventType) {
    case EventType::BKG_REQ:{
      BomJSONExport::jsonExportBookingRequestObject (oStr, iEventStruct);
      break;
    }
    case EventType::CX:
    case EventType::OPT_NOT_4_FD:
    case EventType::OPT_NOT_4_NET:
    case EventType::SKD_CHG:
    case EventType::SNAPSHOT:
    case EventType::RM:
      break;
    case EventType::BRK_PT:
```

```
      BomJSONExport::jsonExportBreakPointObject (oStr, iEventStruct);
      break;
    default:
      break;
    }
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool STDAIR_Service::
  jsonImportConfiguration (const JSONString& iJSONString) const {

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the config holder
    ConfigHolderStruct& lConfigHolder =
      lSTDAIR_ServiceContext.getConfigHolder();

    // Import the JSON string in the configuration holder
    return BomJSONImport::jsonImportConfig (iJSONString, lConfigHolder);
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::
  jsonExportConfiguration () const {

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the config holder
    ConfigHolderStruct& lConfigHolder =
      lSTDAIR_ServiceContext.getConfigHolder();

    // Export the configuration tree in a JSon format
    return lConfigHolder.jsonExport();
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_Service::importINIConfig (const ConfigINIFile& iConfigINIFile) {

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the config holder
    ConfigHolderStruct& lConfigHolder =
      lSTDAIR_ServiceContext.getConfigHolder();

    // Try to import the configuration
    stdair::BomINIImport::importINIConfig (lConfigHolder, iConfigINIFile);
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_Service::importConfigValue (const std::string& iValue,
                                          const std::string& iPath) {

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

```

```
    // Retrieve the config holder
    ConfigHolderStruct& lConfigHolder =
      lSTDAIR_ServiceContext.getConfigHolder();

    // Add the given value to the configuration
    lConfigHolder.addValue (iValue, iPath);
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_Service::updateAirlineFeatures () {

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the config holder
    ConfigHolderStruct& lConfigHolder =
      lSTDAIR_ServiceContext.getConfigHolder();

    // Retrieve the persistent BOM tree root
    BomRoot& lPersistentBomRoot =
      lSTDAIR_ServiceContext.getPersistentBomRoot();

    // Add the given value to the configuration
    lConfigHolder.updateAirlineFeatures (lPersistentBomRoot);
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::list (const AirlineCode_T& iAirlineCode,
                                    const FlightNumber_T& iFlightNumber) const {
    std::ostringstream oStr;

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

    // Retrieve the BOM tree root
    const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

    // Dump the content of the whole BOM tree into the string
    BomDisplay::list (oStr, lCloneBomRoot, iAirlineCode, iFlightNumber);

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::listAirportPairDateRange () const {
    std::ostringstream oStr;

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

    // Retrieve the BOM tree root
    const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

    // Dump the content of the whole BOM tree into the string
    BomDisplay::listAirportPairDateRange (oStr, lCloneBomRoot);

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  bool STDAIR_Service::check (const AirlineCode_T& iAirlineCode,
```

```
                                const FlightNumber_T& iFlightNumber,
                                const stdair::Date_T& iDepartureDate) const {
  std::ostringstream oStr;

  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
  const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

  // Retrieve the BOM tree root
  const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

  // Dump the content of the whole BOM tree into the string
  const FlightDate* lFlightDate_ptr =
    BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot, iAirlineCode,
                                                iFlightNumber,
                                                iDepartureDate);

  return (lFlightDate_ptr != NULL);
}

// ////////////////////////////////////////////////////////////////////
bool STDAIR_Service::check (const stdair::AirportCode_T& ioOrigin,
                                const stdair::AirportCode_T& ioDestination,
                                const stdair::Date_T& ioDepartureDate) const {
  std::ostringstream oStr;

  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
  const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

  // Retrieve the BOM tree root
  const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

  // Dump the content of the whole BOM tree into the string
  stdair::DatePeriodList_T lDatePeriodList;
  BomRetriever::retrieveDatePeriodListFromKeySet  (lCloneBomRoot, ioOrigin,
                                                    ioDestination,
                                                    ioDepartureDate,
                                                    lDatePeriodList);

  return (lDatePeriodList.size() != 0);
}

// ////////////////////////////////////////////////////////////////////
std::string STDAIR_Service::configDisplay () const {

  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
  const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
    *_stdairServiceContext;

  // Retrieve the config holder
  ConfigHolderStruct& lConfigHolder =
    lSTDAIR_ServiceContext.getConfigHolder();

  // Display (dump in the returned string) the configuration.
  return lConfigHolder.describe();

}

// ////////////////////////////////////////////////////////////////////
std::string STDAIR_Service::csvDisplay () const {

  // Retrieve the StdAir service context
  assert (_stdairServiceContext != NULL);
```

```
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
      *_stdairServiceContext;

    // Retrieve the persistent BOM tree root
    const BomRoot& lPersistentBomRoot =
      lSTDAIR_ServiceContext.getPersistentBomRoot();

    // Call the dedicated service
    return csvDisplay (lPersistentBomRoot);
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::csvDisplay (const BomRoot& iBomRoot) const {
    std::ostringstream oStr;

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);

    // Dump the content of the whole BOM tree into the string
    BomDisplay::csvDisplay (oStr, iBomRoot);

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::
  csvDisplay (const stdair::AirlineCode_T& iAirlineCode,
              const stdair::FlightNumber_T& iFlightNumber,
              const stdair::Date_T& iDepartureDate) const {
    std::ostringstream oStr;

    // Retrieve the StdAir service context
    assert (_stdairServiceContext != NULL);
    const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

    // Retrieve the BOM tree root
    const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();

    // Retrieve the flight-date object corresponding to the key
    FlightDate* lFlightDate_ptr =
      BomRetriever::retrieveFlightDateFromKeySet (lCloneBomRoot, iAirlineCode,
                                                  iFlightNumber,
                                                  iDepartureDate);

    // Dump the content of the whole BOM tree into the string
    if (lFlightDate_ptr != NULL) {
      BomDisplay::csvDisplay (oStr, *lFlightDate_ptr);

    } else {
      oStr << "   No flight-date found for the given key: '"
           << iAirlineCode << iFlightNumber << " - " << iDepartureDate << "'";
    }

    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  std::string STDAIR_Service::
  csvDisplay (const TravelSolutionList_T& iTravelSolutionList) const {

    // Dump the content of the whole list of travel solutions into the string
    std::ostringstream oStr;
    BomDisplay::csvDisplay (oStr, iTravelSolutionList);

    return oStr.str();
  }
```

```
00595
00596   // ////////////////////////////////////////////////////////////////////
00597   std::string STDAIR_Service::
00598   csvDisplay (const stdair::AirportCode_T& iOrigin,
00599               const stdair::AirportCode_T& iDestination,
00600               const stdair::Date_T& iDepartureDate) const {
00601     std::ostringstream oStr;
00602
00603     // Retrieve the StdAir service context
00604     assert (_stdairServiceContext != NULL);
00605     const STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;

00606
00607     // Retrieve the BOM tree root
00608     const BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00609
00610     // Retrieve the flight-date object corresponding to the key
00611     DatePeriodList_T lDatePeriodList;
00612     BomRetriever::retrieveDatePeriodListFromKeySet (lCloneBomRoot, iOrigin,
00613                                                     iDestination,
00614                                                     iDepartureDate,
00615                                                     lDatePeriodList);
00616
00617     // Dump the content of the whole BOM tree into the string
00618     if (lDatePeriodList.empty()) {
00619       oStr << "   No fare-rule found for the given key: '"
00620            << iOrigin << "-" << iDestination << " - " << iDepartureDate << "'";
00621     } else {
00622       BomDisplay::csvDisplay (oStr, lDatePeriodList);
00623     }
00624
00625     return oStr.str();
00626   }
00627
00628   // ////////////////////////////////////////////////////////////////////
00629   void STDAIR_Service::finalise() {
00630     // Clean all the objects
00631     FacSupervisor::cleanAll();
00632   }
00633
00634   // ////////////////////////////////////////////////////////////////////
00635   void STDAIR_Service::clonePersistentBom () {
00636
00637     // Retrieve the StdAir service context
00638     assert (_stdairServiceContext != NULL);
00639     STDAIR_ServiceContext& lSTDAIR_ServiceContext = *_stdairServiceContext;
00640
00641     // Clean all the cloned objects
00642     FacSupervisor::instance().cleanCloneBomLayer();
00643
00644     // Init the root of the clone BOM tree
00645     lSTDAIR_ServiceContext.initCloneBomRoot();
00646
00647     // Retrieve the persistent BOM tree root and the clone BOM tree root
00648     const BomRoot& lPersistentBomRoot =
00649       lSTDAIR_ServiceContext.getPersistentBomRoot();
00650     BomRoot& lCloneBomRoot = lSTDAIR_ServiceContext.getCloneBomRoot();
00651
00652     // Call the dedicated service to clone the whole BOM
00653     CmdCloneBomManager::cloneBomRoot (lPersistentBomRoot, lCloneBomRoot);
00654
00655   }
00656
00657 }
```

33.599 stdair/service/STDAIR_ServiceContext.cpp File Reference

33.600 stdair/service/STDAIR_ServiceContext.cpp

```

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
#include <sstream>
// Boost
#if BOOST_VERSION >= 103900
#include <boost/make_shared.hpp>
#else  // BOOST_VERSION >= 103900
#include <boost/shared_ptr.hpp>
#endif // BOOST_VERSION >= 103900
// StdAir
#include <stdair/basic/BasConst_General.hpp>
#include <stdair/bom/BomRoot.hpp>
#include <stdair/factory/FacBom.hpp>
#include <stdair/factory/FacCloneBom.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  STDAIR_ServiceContext::STDAIR_ServiceContext()
    : _cloneBomRoot (NULL),
      _persistentBomRoot (NULL),
      _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
    // Build the BomRoot object
    init();
  }

  // ////////////////////////////////////////////////////////////////////
  STDAIR_ServiceContext::
  STDAIR_ServiceContext (const STDAIR_ServiceContext& iServiceContext)
    : _cloneBomRoot (iServiceContext._cloneBomRoot),
      _persistentBomRoot (iServiceContext._persistentBomRoot),
      _initType (ServiceInitialisationType::NOT_YET_INITIALISED) {
    assert (false);
  }

  // ////////////////////////////////////////////////////////////////////
  STDAIR_ServiceContext::~STDAIR_ServiceContext() {
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::init() {
    //
    initBomRoot();
    initConfigHolder();
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::initBomRoot() {
    _persistentBomRoot = &FacBom<BomRoot>::instance().create();
    initCloneBomRoot();
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::initCloneBomRoot() {
    _cloneBomRoot =
      &FacCloneBom<BomRoot>::instance().clone(*_persistentBomRoot);
  }

  // ////////////////////////////////////////////////////////////////////
  void STDAIR_ServiceContext::initConfigHolder() {
```

```
    _configHolderPtr = boost::make_shared<ConfigHolderStruct> ();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string STDAIR_ServiceContext::shortDisplay() const {
    std::ostringstream oStr;
    oStr << "STDAIR_ServiceContext -- " << _initType
         << " -- DB: " << _dbParams;
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string STDAIR_ServiceContext::display() const {
    std::ostringstream oStr;
    oStr << shortDisplay();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string STDAIR_ServiceContext::describe() const {
    return shortDisplay();
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot& STDAIR_ServiceContext::getPersistentBomRoot() const {
    assert (_persistentBomRoot != NULL);
    return *_persistentBomRoot;
  }

  // ////////////////////////////////////////////////////////////////////
  BomRoot& STDAIR_ServiceContext::getCloneBomRoot() const {
    assert (_cloneBomRoot != NULL);
    return *_cloneBomRoot;
  }

  // ////////////////////////////////////////////////////////////////////
  ConfigHolderStruct& STDAIR_ServiceContext::getConfigHolder() const {
    assert (_configHolderPtr != NULL);
    return *_configHolderPtr;
  }
}
```

33.601 stdair/service/STDAIR_ServiceContext.hpp File Reference

```
#include <string>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/service/ServiceAbstract.hpp>
```

Classes

- class stdair::STDAIR_ServiceContext

  Class holding the context of the Stdair services.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.602 stdair/service/STDAIR_ServiceContext.hpp

```
00001 #ifndef __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00002 #define __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // StdAir
00010 #include <stdair/stdair_basic_types.hpp>
00011 #include <stdair/basic/BasLogParams.hpp>
00012 #include <stdair/basic/BasDBParams.hpp>
00013 #include <stdair/bom/ConfigHolderStruct.hpp>
00014 #include <stdair/basic/ServiceInitialisationType.hpp>
00015 #include <stdair/service/ServiceAbstract.hpp>
00016
00017 namespace stdair {
00018
00020   class BomRoot;
00021
00025   class STDAIR_ServiceContext : public ServiceAbstract {
00029     friend class STDAIR_Service;
00030     friend class FacSTDAIRServiceContext;
00031
00032   private:
00033     // ///////// Getters //////////
00037     BomRoot& getPersistentBomRoot() const;
00038
00042     BomRoot& getCloneBomRoot() const;
00043
00047     ConfigHolderStruct& getConfigHolder() const;
00048
00052     const BasDBParams& getDBParams() const {
00053       return _dbParams;
00054     }
00055
00059     const ServiceInitialisationType& getServiceInitialisationType() const {
00060       return _initType;
00061     }
00062
00063
00064   private:
00065     // ///////// Setters //////////
00069     void setDBParams (const BasDBParams& iDBParams) {
00070       _dbParams = iDBParams;
00071     }
00072
00076     void setServiceInitialisationType (const ServiceInitialisationType& iSIT) {
00077       _initType = iSIT;
00078     }
00079
00080
00081   private:
00082     // ///////// Display Methods //////////
00086     const std::string shortDisplay() const;
00087
00091     const std::string display() const;
00092
00096     const std::string describe() const;
00097
00098
00099   private:
00100     // /////// Construction / initialisation ////////
00104     STDAIR_ServiceContext();
00105
```

```
    STDAIR_ServiceContext (const STDAIR_ServiceContext&);

    ~STDAIR_ServiceContext();

    void init();

    void initBomRoot();

    void initCloneBomRoot();

    void initConfigHolder();

  private:
    // ///////////// Children ///////////
    BomRoot* _cloneBomRoot;

    BomRoot* _persistentBomRoot;

    ConfigHolderPtr_T _configHolderPtr;

    BasDBParams _dbParams;

    ServiceInitialisationType _initType;
  };

}
#endif // __STDAIR_SVC_STDAIRSERVICECONTEXT_HPP
```

33.603 stdair/stdair_basic_types.hpp File Reference

```
#include <string>
#include <list>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::string stdair::LocationCode_T
- typedef unsigned long int stdair::Distance_T
- typedef LocationCode_T stdair::AirportCode_T
- typedef LocationCode_T stdair::CityCode_T
- typedef std::string stdair::KeyDescription_T
- typedef std::string stdair::AirlineCode_T
- typedef unsigned short stdair::FlightNumber_T
- typedef unsigned short stdair::TableID_T
- typedef std::string stdair::CabinCode_T
- typedef std::string stdair::FamilyCode_T
- typedef std::string stdair::PolicyCode_T
- typedef std::string stdair::NestingStructureCode_T
- typedef std::string stdair::NestingNodeCode_T
- typedef std::string stdair::ClassCode_T
- typedef unsigned long stdair::Identity_T
- typedef std::string stdair::TripType_T
- typedef double stdair::MonetaryValue_T
- typedef double stdair::RealNumber_T
- typedef double stdair::Percentage_T
- typedef double stdair::PriceValue_T
- typedef double stdair::YieldValue_T
- typedef std::string stdair::PriceCurrency_T
- typedef double stdair::Revenue_T
- typedef double stdair::Multiplier_T
- typedef double stdair::NbOfSeats_T
- typedef unsigned int stdair::Count_T
- typedef short stdair::PartySize_T
- typedef double stdair::NbOfRequests_T
- typedef NbOfRequests_T stdair::NbOfBookings_T
- typedef NbOfRequests_T stdair::NbOfCancellations_T
- typedef unsigned short stdair::NbOfTravelSolutions_T
- typedef std::string stdair::ClassList_String_T
- typedef unsigned short stdair::NbOfSegments_T
- typedef unsigned short stdair::NbOfAirlines_T
- typedef double stdair::Availability_T
- typedef double stdair::Fare_T

- typedef bool stdair::Flag_T
- typedef unsigned int stdair::UnsignedIndex_T
- typedef unsigned int stdair::NbOfClasses_T
- typedef unsigned int stdair::NbOfFareFamilies_T
- typedef std::string stdair::Filename_T
- typedef std::string stdair::FileAddress_T
- typedef float stdair::ProgressPercentage_T

33.604 stdair/stdair_basic_types.hpp

```
#ifndef __STDAIR_STDAIR_BASIC_TYPES_HPP
#define __STDAIR_STDAIR_BASIC_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <list>

namespace stdair {

  // ////////////////////////////////////////////////////////////////////
  // Basic types
  typedef std::string LocationCode_T;

  typedef unsigned long int Distance_T;

  typedef LocationCode_T AirportCode_T;

  typedef LocationCode_T CityCode_T;

  typedef std::string KeyDescription_T;

  typedef std::string AirlineCode_T;

  typedef unsigned short FlightNumber_T;

  typedef unsigned short TableID_T;

  typedef std::string CabinCode_T;

  typedef std::string FamilyCode_T;

  typedef std::string PolicyCode_T;

  typedef std::string NestingStructureCode_T;

  typedef std::string NestingNodeCode_T;

  typedef std::string ClassCode_T;

  typedef unsigned long Identity_T;

  typedef std::string TripType_T;

  typedef double MonetaryValue_T;

  typedef double RealNumber_T;

  typedef double Percentage_T;

  typedef double PriceValue_T;

  typedef double YieldValue_T;

  typedef std::string PriceCurrency_T;

  typedef double Revenue_T;

  typedef double Multiplier_T;

  typedef double NbOfSeats_T;

  typedef unsigned int Count_T;
```

```
00096
00098   typedef short PartySize_T;
00099
00101   typedef double NbOfRequests_T;
00102
00104   typedef NbOfRequests_T NbOfBookings_T;
00105
00107   typedef NbOfRequests_T NbOfCancellations_T;
00108
00111   typedef unsigned short NbOfTravelSolutions_T;
00112
00114   typedef std::string ClassList_String_T;
00115
00117   typedef unsigned short NbOfSegments_T;
00118
00120   typedef unsigned short NbOfAirlines_T;
00121
00123   typedef double Availability_T;
00124
00126   typedef double Fare_T;
00127
00129   typedef bool Flag_T;
00130
00132   typedef unsigned int UnsignedIndex_T;
00133
00135   typedef unsigned int NbOfClasses_T;
00136
00138   typedef unsigned int NbOfFareFamilies_T;
00139
00140   // ///////////// Technical ///////////////
00144   typedef std::string Filename_T;
00145
00148   typedef std::string FileAddress_T;
00149
00152   typedef float ProgressPercentage_T;
00153
00154 }
00155 #endif // __STDAIR_STDAIR_BASIC_TYPES_HPP
```

33.605 stdair/stdair_date_time_types.hpp File Reference

```
#include <string>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::posix_time::time_duration stdair::Duration_T
- typedef boost::gregorian::date stdair::Date_T
- typedef boost::posix_time::time_duration stdair::Time_T
- typedef boost::posix_time::ptime stdair::DateTime_T
- typedef boost::gregorian::date_period stdair::DatePeriod_T
- typedef std::string stdair::DOW_String_T
- typedef boost::gregorian::date_duration stdair::DateOffset_T
- typedef int stdair::DayDuration_T
- typedef bool stdair::SaturdayStay_T
- typedef long int stdair::IntDuration_T
- typedef long long int stdair::LongDuration_T
- typedef float stdair::FloatDuration_T

33.606 stdair/stdair_date_time_types.hpp

```
#ifndef __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
#define __STDAIR_STDAIR_DATE_TIME_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
// Boost (Extended STL)
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>

namespace stdair {

  // //////// Type definitions /////////
  typedef boost::posix_time::time_duration Duration_T;

  typedef boost::gregorian::date Date_T;

  typedef boost::posix_time::time_duration Time_T;

  typedef boost::posix_time::ptime DateTime_T;

  typedef boost::gregorian::date_period DatePeriod_T;

  typedef std::string DOW_String_T;

  typedef boost::gregorian::date_duration DateOffset_T;

  typedef int DayDuration_T;

  typedef bool SaturdayStay_T;

  typedef long int IntDuration_T;

  typedef long long int LongDuration_T;

  typedef float FloatDuration_T;

}
#endif // __STDAIR_STDAIR_DATE_TIME_TYPES_HPP
```

33.607 stdair/stdair_db.hpp File Reference

```
#include <string>
```

Namespaces

- namespace soci
- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef soci::session stdair::DBSession_T
- typedef soci::statement stdair::DBRequestStatement_T
- typedef std::string stdair::DBConnectionName_T

33.608 stdair/stdair_db.hpp

```
#ifndef __STDAIR_STDAIR_DB_HPP
#define __STDAIR_STDAIR_DB_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

// Forward declarations
namespace soci {
  class session;
  class statement;
}

namespace stdair {

  // //////// Type definitions /////////
  typedef soci::session DBSession_T;

  typedef soci::statement DBRequestStatement_T;

  typedef std::string DBConnectionName_T;

}
#endif // __STDAIR_STDAIR_DB_HPP
```

33.609 stdair/stdair_demand_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/date_time/gregorian/gregorian.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/tuple/tuple.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef bool stdair::ChangeFees_T
- typedef bool stdair::NonRefundable_T
- typedef double stdair::SaturdayStayRatio_T
- typedef double stdair::ChangeFeesRatio_T
- typedef double stdair::NonRefundableRatio_T
- typedef double stdair::Disutility_T
- typedef std::string stdair::PassengerType_T
- typedef std::string stdair::DistributionPatternId_T
- typedef std::string stdair::CancellationRateCurveId_T
- typedef std::string stdair::AirlinePreferenceId_T
- typedef std::pair< Percentage_T, Percentage_T > stdair::CancellationNoShowRatePair_T
- typedef std::string stdair::CharacteristicsPatternId_T
- typedef std::string stdair::CharacteristicsIndex_T
- typedef double stdair::WTP_T
- typedef boost::tuples::tuple< double, WTP_T > stdair::CharacteristicsWTP_tuple_T
- typedef std::pair< WTP_T, MeanStdDevPair_T > stdair::WTPDemandPair_T
- typedef NbOfRequests_T stdair::NbOfNoShows_T
- typedef double stdair::MatchingIndicator_T
- typedef std::string stdair::DemandStreamKeyStr_T
- typedef std::string stdair::ChannelLabel_T
- typedef std::string stdair::FrequentFlyer_T
- typedef std::string stdair::RequestStatus_T

- typedef std::map< Identity_T, Identity_T > stdair::BookingTSIDMap_T
- typedef std::pair< CabinCode_T, ClassCode_T > stdair::CabinClassPair_T
- typedef std::list< CabinClassPair_T > stdair::CabinClassPairList_T
- typedef double stdair::ProportionFactor_T
- typedef std::list< ProportionFactor_T > stdair::ProportionFactorList_T
- typedef std::string stdair::OnDString_T
- typedef std::list< OnDString_T > stdair::OnDStringList_T

33.610 stdair/stdair_demand_types.hpp

```
00001 #ifndef __STDAIR_STDAIR_DEMAND_TYPES_HPP
00002 #define __STDAIR_STDAIR_DEMAND_TYPES_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 #include <vector>
00010 #include <map>
00011 // Boost Random
00012 #include <boost/random/linear_congruential.hpp>
00013 #include <boost/random/uniform_real.hpp>
00014 #include <boost/random/variate_generator.hpp>
00015 // Boost (Extended STL)
00016 #include <boost/date_time/gregorian/gregorian.hpp>
00017 #include <boost/date_time/posix_time/posix_time.hpp>
00018 #include <boost/tuple/tuple.hpp>
00019 // StdAir
00020 #include <stdair/stdair_basic_types.hpp>
00021 #include <stdair/stdair_maths_types.hpp>
00022 #include <stdair/stdair_inventory_types.hpp>
00023
00024
00025 namespace stdair {
00026
00027   // //////// Type definitions /////////
00029   typedef bool ChangeFees_T;
00030
00032   typedef bool NonRefundable_T;
00033
00035   typedef bool SaturdayStay_T;
00036
00039   typedef double SaturdayStayRatio_T;
00040
00043   typedef double ChangeFeesRatio_T;
00044
00047   typedef double NonRefundableRatio_T;
00048
00050   typedef double Disutility_T;
00051
00054   typedef std::string PassengerType_T;
00055
00057   typedef std::string DistributionPatternId_T;
00058
00060   typedef std::string CancellationRateCurveId_T;
00061
00063   typedef std::string AirlinePreferenceId_T;
00064
00066   typedef std::pair<Percentage_T, Percentage_T> CancellationNoShowRatePair_T;
00067
00070   typedef std::string CharacteristicsPatternId_T;
00071
00073   typedef std::string CharacteristicsIndex_T;
00074
00076   typedef double WTP_T;
00077
00079   typedef boost::tuples::tuple<double, WTP_T> CharacteristicsWTP_tuple_T;
00080
00082   typedef std::pair<WTP_T, MeanStdDevPair_T> WTPDemandPair_T;
00083
00085   typedef NbOfRequests_T NbOfCancellations_T;
00086
00088   typedef NbOfRequests_T NbOfNoShows_T;
00089
```

```
00091   typedef double MatchingIndicator_T;
00092
00094   typedef std::string DemandStreamKeyStr_T;
00095
00097   typedef std::string ChannelLabel_T;
00098
00100   typedef std::string FrequentFlyer_T;
00101
00104   typedef std::string RequestStatus_T;
00105
00107   typedef std::map<Identity_T, Identity_T> BookingTSIDMap_T;
00108
00110   typedef std::pair<CabinCode_T, ClassCode_T> CabinClassPair_T;
00111
00113   typedef std::list<CabinClassPair_T> CabinClassPairList_T;
00114
00116   typedef double ProportionFactor_T;
00117
00119   typedef std::list<ProportionFactor_T> ProportionFactorList_T;
00120
00122   typedef std::string OnDString_T;
00123
00125   typedef std::list<OnDString_T> OnDStringList_T;
00126
00127 }
00128 #endif // __STDAIR_STDAIR_DEMAND_TYPES_HPP
```

33.611 stdair/stdair_event_types.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::string stdair::EventName_T
- typedef double stdair::NbOfEvents_T
- typedef std::string stdair::EventGeneratorKey_T

33.612 stdair/stdair_event_types.hpp

```
#ifndef __STDAIR_STDAIR_EVENT_TYPES_HPP
#define __STDAIR_STDAIR_EVENT_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

  // //////// Type definitions /////////
  typedef std::string EventName_T;

  typedef double NbOfEvents_T;

  typedef std::string EventGeneratorKey_T;

}
#endif // __STDAIR_STDAIR_EVENT_TYPES_HPP
```

33.613 stdair/stdair_exceptions.hpp File Reference

```
#include <string>
```

Classes

- class stdair::RootException

  Root of the stdair exceptions.

- class stdair::FileNotFoundException
- class stdair::NonInitialisedLogServiceException
- class stdair::NonInitialisedServiceException
- class stdair::NonInitialisedContainerException
- class stdair::NonInitialisedRelationShipException
- class stdair::MemoryAllocationException
- class stdair::ObjectLinkingException
- class stdair::DocumentNotFoundException
- class stdair::ParserException
- class stdair::SerialisationException
- class stdair::KeyNotFoundException
- class stdair::CodeConversionException
- class stdair::CodeDuplicationException
- class stdair::KeyDuplicationException
- class stdair::ObjectCreationgDuplicationException
- class stdair::ObjectNotFoundException
- class stdair::ParsingFileFailedException
- class stdair::SQLDatabaseException
- class stdair::NonInitialisedDBSessionManagerException
- class stdair::SQLDatabaseConnectionImpossibleException
- class stdair::EventException
- class stdair::SimpleNestingStructException
- class stdair::BookingClassListEmptyInNestingStructException

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.614 stdair/stdair_exceptions.hpp

```
#ifndef __STDAIR_STDAIR_EXCEPTIONS_HPP
#define __STDAIR_STDAIR_EXCEPTIONS_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

  class RootException : public std::exception {
  public:
    RootException (const std::string& iWhat) : _what (iWhat) {}
    RootException() : _what ("No further details") {}

    virtual ~RootException() throw() {}

    const char* what() const throw() {
      return _what.c_str();
    }

  protected:
    std::string _what;
  };

  class FileNotFoundException : public RootException {
  public:
    FileNotFoundException (const std::string& iWhat) : RootException (iWhat) {}
  };

  class NonInitialisedLogServiceException : public RootException {
  public:
    NonInitialisedLogServiceException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class NonInitialisedServiceException : public RootException {
  public:
    NonInitialisedServiceException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class NonInitialisedContainerException : public RootException {
  public:
    NonInitialisedContainerException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class NonInitialisedRelationShipException : public RootException {
  public:
    NonInitialisedRelationShipException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class MemoryAllocationException : public RootException {
  public:
    MemoryAllocationException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class ObjectLinkingException : public RootException {
  public:
    ObjectLinkingException (const std::string& iWhat) : RootException (iWhat) {}
  };
```

```
00102
00104   class DocumentNotFoundException : public RootException {
00105   public:
00107     DocumentNotFoundException (const std::string& iWhat)
00108       : RootException (iWhat) {}
00109   };
00110
00112   class ParserException : public RootException {
00113   public:
00115     ParserException (const std::string& iWhat) : RootException (iWhat) {}
00116   };
00117
00119   class SerialisationException : public RootException {
00120   public:
00122     SerialisationException (const std::string& iWhat) : RootException (iWhat) {}
00123   };
00124
00126   class KeyNotFoundException : public RootException {
00127   public:
00129     KeyNotFoundException (const std::string& iWhat) : RootException (iWhat) {}
00130   };
00131
00133   class CodeConversionException : public ParserException {
00134   public:
00136     CodeConversionException (const std::string& iWhat)
00137       : ParserException (iWhat) {}
00138   };
00139
00141   class CodeDuplicationException : public ParserException {
00142   public:
00144     CodeDuplicationException (const std::string& iWhat)
00145       : ParserException(iWhat) {}
00146   };
00147
00149   class KeyDuplicationException : public ParserException {
00150   public:
00152     KeyDuplicationException (const std::string& iWhat)
00153       : ParserException(iWhat) {}
00154   };
00155
00157   class ObjectCreationgDuplicationException : public ParserException {
00158   public:
00160     ObjectCreationgDuplicationException (const std::string& iWhat)
00161       : ParserException (iWhat) {}
00162   };
00163
00165   class ObjectNotFoundException : public RootException {
00166   public:
00168     ObjectNotFoundException (const std::string& iWhat)
00169       : RootException (iWhat) {}
00170   };
00171
00173   class ParsingFileFailedException : public ParserException {
00174   public:
00176     ParsingFileFailedException (const std::string& iWhat)
00177       : ParserException (iWhat) {}
00178   };
00179
00181   class SQLDatabaseException : public RootException {
00182   public:
00184     SQLDatabaseException (const std::string& iWhat) : RootException (iWhat) {}
00185   };
00186
00188   class NonInitialisedDBSessionManagerException : public RootException {
00189   public:
00191     NonInitialisedDBSessionManagerException (const std::string& iWhat)
00192       : RootException (iWhat) {}
```

```
  };

  class SQLDatabaseConnectionImpossibleException : public SQLDatabaseException {
  public:
    SQLDatabaseConnectionImpossibleException (const std::string& iWhat)
      : SQLDatabaseException (iWhat) {}
  };

  class EventException : public RootException {
  public:
    EventException (const std::string& iWhat) : RootException (iWhat) {}
  };

  class SimpleNestingStructException : public RootException {
  public:
    SimpleNestingStructException (const std::string& iWhat)
      : RootException (iWhat) {}
  };

  class BookingClassListEmptyInNestingStructException :
    public SimpleNestingStructException {
  public:
    BookingClassListEmptyInNestingStructException (const std::string& iWhat)
      : SimpleNestingStructException (iWhat) {}
  };

}
#endif // __STDAIR_STDAIR_EXCEPTIONS_HPP
```

33.615 stdair/stdair_fare_types.hpp File Reference

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef double stdair::NbOfFareRules_T

33.616 stdair/stdair_fare_types.hpp

```
#ifndef __STDAIR_STDAIR_FARE_TYPES_HPP
#define __STDAIR_STDAIR_FARE_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////

namespace stdair {

  // //////// Type definitions /////////
  typedef double NbOfFareRules_T;

}
#endif // __STDAIR_STDAIR_FARE_TYPES_HPP
```

33.617 stdair/stdair_file.hpp File Reference

```
#include <string>
#include <boost/utility.hpp>
#include <stdair/stdair_basic_types.hpp>
```

Classes

- class stdair::RootFilePath

  Root of the input and output files.

- class stdair::InputFilePath
- class stdair::ScheduleFilePath
- class stdair::ODFilePath
- class stdair::FRAT5FilePath
- class stdair::FFDisutilityFilePath
- class stdair::ConfigINIFile

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.618 stdair/stdair_file.hpp

```
00001 #ifndef __STDAIR_STDAIR_FILE_HPP
00002 #define __STDAIR_STDAIR_FILE_HPP
00003
00004 // ////////////////////////////////////////////////////////////////////
00005 // Import section
00006 // ////////////////////////////////////////////////////////////////////
00007 // STL
00008 #include <string>
00009 // Boost
00010 #include <boost/utility.hpp>
00011 // StdAir
00012 #include <stdair/stdair_basic_types.hpp>
00013
00014 namespace stdair {
00015
00022   class RootFilePath {
00023   public:
00027     RootFilePath (const Filename_T& iFilename) :
00028       _filename (iFilename) {}
00032     RootFilePath () : _filename ("MyFilename") {}
00033
00037     virtual ~RootFilePath() {}
00038
00042     const char * name() const {
00043       return _filename.c_str();
00044     }
00045
00046   protected:
00050     const Filename_T _filename;
00051   };
00052
00054   class InputFilePath : public RootFilePath {
00055   public:
00057     InputFilePath (const Filename_T& iFilename) :
00058       RootFilePath (iFilename) {}
00059   };
00060
00064   class ScheduleFilePath : public InputFilePath {
00065   public:
00069     explicit ScheduleFilePath (const Filename_T& iFilename)
00070       : InputFilePath (iFilename) {}
00071   };
00072
00076   class ODFilePath : public InputFilePath {
00077   public:
00081     explicit ODFilePath (const Filename_T& iFilename)
00082       : InputFilePath (iFilename) {}
00083   };
00084
00088   class FRAT5FilePath : public InputFilePath {
00089   public:
00093     explicit FRAT5FilePath (const Filename_T& iFilename)
00094       : InputFilePath (iFilename) {}
00095   };
00096
00100   class FFDisutilityFilePath : public InputFilePath {
00101   public:
00105     explicit FFDisutilityFilePath (const Filename_T& iFilename)
00106       : InputFilePath (iFilename) {}
00107   };
00108
00112   class ConfigINIFile : public InputFilePath {
00113   public:
00117     explicit ConfigINIFile (const Filename_T& iFilename)
00118       : InputFilePath (iFilename) {}
```

```

  };

}
#endif // __STDAIR_STDAIR_FILE_HPP
```

33.619 stdair/stdair_inventory_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <list>
#include <boost/multi_array.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::string stdair::NetworkID_T
- typedef std::vector< AirlineCode_T > stdair::AirlineCodeList_T
- typedef std::vector< ClassList_String_T > stdair::ClassList_StringList_T
- typedef std::vector< ClassCode_T > stdair::ClassCodeList_T
- typedef unsigned short stdair::SubclassCode_T
- typedef std::string stdair::FlightPathCode_T
- typedef std::map< CabinCode_T, ClassList_String_T > stdair::CabinBookingClassMap_T
- typedef std::string stdair::CurveKey_T
- typedef double stdair::CabinCapacity_T
- typedef double stdair::NbOfFlightDates_T
- typedef double stdair::CommittedSpace_T
- typedef double stdair::UPR_T
- typedef double stdair::BookingLimit_T
- typedef double stdair::AuthorizationLevel_T
- typedef double stdair::CapacityAdjustment_T
- typedef double stdair::BlockSpace_T
- typedef bool stdair::AvailabilityStatus_T
- typedef std::vector< Availability_T > stdair::BucketAvailabilities_T
- typedef double stdair::NbOfYields_T
- typedef double stdair::NbOfInventoryControlRules_T
- typedef bool stdair::CensorshipFlag_T
- typedef short stdair::DTD_T
- typedef short stdair::DCP_T
- typedef std::list< DCP_T > stdair::DCPList_T
- typedef std::map< DTD_T, RealNumber_T > stdair::DTDFratMap_T
- typedef std::map< FloatDuration_T, float > stdair::DTDProbMap_T
- typedef std::vector< CensorshipFlag_T > stdair::CensorshipFlagList_T
- typedef double stdair::BookingRatio_T

- typedef double stdair::Yield_T
- typedef unsigned int stdair::YieldLevel_T
- typedef std::map< YieldLevel_T, MeanStdDevPair_T > stdair::YieldLevelDemandMap_T
- typedef std::pair< Yield_T, MeanStdDevPair_T > stdair::YieldDemandPair_T
- typedef double stdair::BidPrice_T
- typedef std::vector< BidPrice_T > stdair::BidPriceVector_T
- typedef unsigned int stdair::SeatIndex_T
- typedef std::string stdair::ControlMode_T
- typedef double stdair::OverbookingRate_T
- typedef double stdair::ProtectionLevel_T
- typedef std::vector< double > stdair::EmsrValueList_T
- typedef std::vector< double > stdair::BookingLimitVector_T
- typedef std::vector< double > stdair::ProtectionLevelVector_T
- typedef boost::multi_array< double, 2 > stdair::SnapshotBlock_T
- typedef SnapshotBlock_T::index_range stdair::SnapshotBlockRange_T
- typedef SnapshotBlock_T::array_view< 1 >::type stdair::SegmentCabinDTDSnapshotView_T
- typedef SnapshotBlock_T::array_view< 2 >::type stdair::SegmentCabinDTDRangeSnapshotView_-T
- typedef SnapshotBlock_T::const_array_view< 1 >::type stdair::ConstSegmentCabinDTDSnapshotView_-T
- typedef SnapshotBlock_T::const_array_view< 2 >::type stdair::ConstSegmentCabinDTDRangeSnapshotView_-T
- typedef unsigned short stdair::SegmentDataID_T
- typedef unsigned short stdair::LegDataID_T
- typedef unsigned short stdair::ClassIndex_T

33.620 stdair/stdair_inventory_types.hpp

```
#ifndef __STDAIR_STDAIR_INVENTORY_TYPES_HPP
#define __STDAIR_STDAIR_INVENTORY_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
#include <map>
#include <list>
// BOOST
#include <boost/multi_array.hpp>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_date_time_types.hpp>

namespace stdair {

  // //////// Type definitions /////////
  typedef std::string NetworkID_T;

  typedef std::vector<AirlineCode_T> AirlineCodeList_T;

  typedef std::vector<ClassList_String_T> ClassList_StringList_T;

  typedef std::vector<ClassCode_T> ClassCodeList_T;

  typedef unsigned short SubclassCode_T;

  typedef std::string FlightPathCode_T;

  typedef std::map<CabinCode_T, ClassList_String_T> CabinBookingClassMap_T;

  typedef std::string CurveKey_T;

  typedef double CabinCapacity_T;

  typedef double NbOfFlightDates_T;

  typedef double CommittedSpace_T;

  typedef double UPR_T;

  typedef double BookingLimit_T;

  typedef double AuthorizationLevel_T;

  typedef double CapacityAdjustment_T;

  typedef double BlockSpace_T;

  typedef bool AvailabilityStatus_T;

  typedef std::vector<Availability_T> BucketAvailabilities_T;

  typedef double NbOfYields_T;

  typedef double NbOfInventoryControlRules_T;

  typedef bool CensorshipFlag_T;

  typedef short DTD_T;

```

```
  typedef short DCP_T;

  typedef std::list<DCP_T> DCPList_T;

  typedef std::map<DTD_T, RealNumber_T> DTDFratMap_T;

  typedef std::map<FloatDuration_T, float> DTDProbMap_T;

  typedef std::vector<CensorshipFlag_T> CensorshipFlagList_T;

  typedef double BookingRatio_T;

  typedef double Yield_T;

  typedef unsigned int YieldLevel_T;

  typedef std::map<YieldLevel_T, MeanStdDevPair_T> YieldLevelDemandMap_T;

  typedef std::pair<Yield_T, MeanStdDevPair_T> YieldDemandPair_T;

  typedef double BidPrice_T;

  typedef std::vector<BidPrice_T> BidPriceVector_T;

  typedef unsigned int SeatIndex_T;

  typedef std::string ControlMode_T;

  typedef double OverbookingRate_T;

  typedef double BookingLimit_T;

  typedef double ProtectionLevel_T;

  typedef std::vector<double> EmsrValueList_T;

  typedef std::vector<double> BookingLimitVector_T;

  typedef std::vector<double> ProtectionLevelVector_T;

  typedef boost::multi_array<double, 2> SnapshotBlock_T;

  typedef SnapshotBlock_T::index_range SnapshotBlockRange_T;

  typedef SnapshotBlock_T::array_view<1>::type SegmentCabinDTDSnapshotView_T;

  typedef SnapshotBlock_T::array_view<2>::type
SegmentCabinDTDRangeSnapshotView_T;

  typedef SnapshotBlock_T::const_array_view<1>::type
ConstSegmentCabinDTDSnapshotView_T;

  typedef SnapshotBlock_T::const_array_view<2>::type
ConstSegmentCabinDTDRangeSnapshotView_T;

  typedef unsigned short SegmentDataID_T;

  typedef unsigned short LegDataID_T;

  typedef unsigned short ClassIndex_T;

}
#endif // __STDAIR_STDAIR_INVENTORY_TYPES_HPP
```

33.621 stdair/stdair_json.hpp File Reference

```
#include <string>
```

Classes

- class stdair::JSONString

  JSON-formatted string.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.622 stdair/stdair_json.hpp

```
#ifndef __STDAIR_STDAIR_JSON_HPP
#define __STDAIR_STDAIR_JSON_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

  class JSONString {
  public:
    explicit JSONString (const std::string& iJsonString) :
      _jsonString (iJsonString) {}
    explicit JSONString () : _jsonString ("") {}

    virtual ~JSONString() {}

    const std::string& getString() const {
      return _jsonString;
    }

  protected:
    std::string _jsonString;
  };

}
#endif // __STDAIR_STDAIR_JSON_HPP
```

33.623 stdair/stdair_log.hpp File Reference

```
#include <string>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

- namespace stdair::LOG

Enumerations

- enum stdair::LOG::EN_LogLevel {

  stdair::LOG::CRITICAL = 0, stdair::LOG::ERROR, stdair::LOG::NOTIFICATION, stdair::LOG::WARNING,

  stdair::LOG::DEBUG, stdair::LOG::VERBOSE, stdair::LOG::LAST_VALUE }

Variables

- static const std::string stdair::LOG::_logLevels [LAST_VALUE]

33.624 stdair/stdair_log.hpp

```
#ifndef __STDAIR_STDAIR_LOG_HPP
#define __STDAIR_STDAIR_LOG_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>

namespace stdair {

  // Forward declarations
  class STDAIR_Service;

  // /////////////// Log /////////////
  namespace LOG {
    typedef enum {
      CRITICAL = 0,
      ERROR,
      NOTIFICATION,
      WARNING,
      DEBUG,
      VERBOSE,
      LAST_VALUE
    } EN_LogLevel;

    static const std::string _logLevels[LAST_VALUE] =
      {"C", "E", "N", "W", "D", "V"};
  }

}
#endif // __STDAIR_STDAIR_LOG_HPP
```

33.625 stdair/stdair_maths_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/variate_generator.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef unsigned int stdair::ReplicationNumber_T
- typedef unsigned long int stdair::ExponentialSeed_T
- typedef unsigned long int stdair::UniformSeed_T
- typedef unsigned long int stdair::RandomSeed_T
- typedef boost::minstd_rand stdair::BaseGenerator_T
- typedef boost::uniform_real stdair::UniformDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, UniformDistribution_T > stdair::UniformGenerator_T
- typedef boost::normal_distribution stdair::NormalDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, NormalDistribution_T > stdair::NormalGenerator_T
- typedef boost::exponential_distribution stdair::ExponentialDistribution_T
- typedef boost::variate_generator< BaseGenerator_T &, ExponentialDistribution_T > stdair::ExponentialGenerator_T
- typedef double stdair::MeanValue_T
- typedef double stdair::StdDevValue_T
- typedef std::pair< MeanValue_T, StdDevValue_T > stdair::MeanStdDevPair_T
- typedef std::vector< MeanStdDevPair_T > stdair::MeanStdDevPairVector_T
- typedef float stdair::Probability_T

33.626 stdair/stdair_maths_types.hpp

```
#ifndef __STDAIR_STDAIR_MATHS_TYPES_HPP
#define __STDAIR_STDAIR_MATHS_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
#include <map>
// Boost Random
#include <boost/random/linear_congruential.hpp>
#include <boost/random/uniform_real.hpp>
#include <boost/random/normal_distribution.hpp>
#include <boost/random/exponential_distribution.hpp>
#include <boost/random/variate_generator.hpp>

namespace stdair {

  // //////// Type definitions /////////
  typedef unsigned int ReplicationNumber_T;

  typedef unsigned long int ExponentialSeed_T;

  typedef unsigned long int UniformSeed_T;

  typedef unsigned long int RandomSeed_T;

  typedef boost::minstd_rand BaseGenerator_T;

  typedef boost::uniform_real<> UniformDistribution_T;

  typedef boost::variate_generator<BaseGenerator_T&,
                                   UniformDistribution_T> UniformGenerator_T;

  typedef boost::normal_distribution<> NormalDistribution_T;

  typedef boost::variate_generator<BaseGenerator_T&,
                                   NormalDistribution_T> NormalGenerator_T;

  typedef boost::exponential_distribution<> ExponentialDistribution_T;


  typedef boost::variate_generator<BaseGenerator_T&,
                                   ExponentialDistribution_T>
ExponentialGenerator_T;

  typedef double MeanValue_T;

  typedef double StdDevValue_T;

  typedef std::pair<MeanValue_T, StdDevValue_T> MeanStdDevPair_T;

  typedef std::vector<MeanStdDevPair_T> MeanStdDevPairVector_T;

  typedef float Probability_T;

}
#endif // __STDAIR_STDAIR_MATHS_TYPES_HPP
```

33.627 stdair/stdair_rm_types.hpp File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef std::string stdair::ForecasterMode_T
- typedef short stdair::HistoricalDataLimit_T
- typedef std::string stdair::OptimizerMode_T
- typedef NbOfBookings_T stdair::PolicyDemand_T
- typedef std::vector< double > stdair::GeneratedDemandVector_T
- typedef std::vector< GeneratedDemandVector_T > stdair::GeneratedDemandVectorHolder_T
- typedef double stdair::SellupProbability_T
- typedef std::vector< NbOfRequests_T > stdair::UncDemVector_T
- typedef std::vector< NbOfBookings_T > stdair::BookingVector_T
- typedef double stdair::FRAT5_T
- typedef std::map< const DTD_T, FRAT5_T > stdair::FRAT5Curve_T
- typedef std::map< const DTD_T, double > stdair::FFDisutilityCurve_T
- typedef std::map< const DTD_T, double > stdair::SellUpCurve_T
- typedef std::map< const DTD_T, double > stdair::DispatchingCurve_T
- typedef std::map< BookingClass *, SellUpCurve_T > stdair::BookingClassSellUpCurveMap_T
- typedef std::map< BookingClass *, DispatchingCurve_T > stdair::BookingClassDispatchingCurveMap_T
- typedef std::map< const Yield_T, double > stdair::YieldDemandMap_T
- typedef unsigned int stdair::NbOfSamples_T

33.628 stdair/stdair_rm_types.hpp

```
#ifndef __STDAIR_STDAIR_RM_TYPES_HPP
#define __STDAIR_STDAIR_RM_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <string>
#include <vector>
#include <map>
// StdAir
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_inventory_types.hpp>

namespace stdair {
  // Forward declarations.
  class BookingClass;

  // //////// Type definitions /////////
  typedef std::string ForecasterMode_T;

  typedef short HistoricalDataLimit_T;

  typedef std::string OptimizerMode_T;

  typedef NbOfBookings_T PolicyDemand_T;

  typedef std::vector<double> GeneratedDemandVector_T;

  typedef std::vector<GeneratedDemandVector_T> GeneratedDemandVectorHolder_T;

  typedef double SellupProbability_T;

  typedef std::vector<NbOfRequests_T> UncDemVector_T;

  typedef std::vector<NbOfBookings_T> BookingVector_T;

  typedef double FRAT5_T;

  typedef std::map<const DTD_T, FRAT5_T> FRAT5Curve_T;

  typedef std::map<const DTD_T, double> FFDisutilityCurve_T;

  typedef std::map<const DTD_T, double> SellUpCurve_T;

  typedef std::map<const DTD_T, double> DispatchingCurve_T;

  typedef std::map<BookingClass*, SellUpCurve_T> BookingClassSellUpCurveMap_T;

  typedef std::map<BookingClass*, DispatchingCurve_T>
BookingClassDispatchingCurveMap_T;

  typedef std::map<const Yield_T, double> YieldDemandMap_T;

  typedef double Revenue_T;

  typedef unsigned int NbOfSamples_T;

}
#endif // __STDAIR_STDAIR_RM_TYPES_HPP
```

33.629 stdair/STDAIR_Service.hpp File Reference

```
#include <string>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_service_types.hpp>
#include <stdair/stdair_file.hpp>
#include <stdair/basic/BasLogParams.hpp>
#include <stdair/basic/BasDBParams.hpp>
#include <stdair/basic/ServiceInitialisationType.hpp>
#include <stdair/bom/TravelSolutionTypes.hpp>
#include <stdair/bom/ConfigHolderStruct.hpp>
#include <stdair/service/STDAIR_ServiceContext.hpp>
```

Classes

- class stdair::STDAIR_Service

  Interface for the STDAIR Services.

Namespaces

- namespace stdair

  Handle on the StdAir library context.

33.630 stdair/STDAIR_Service.hpp

```
00001
00012 #ifndef __STDAIR_STDAIR_HPP
00013 #define __STDAIR_STDAIR_HPP
00014
00015 // ////////////////////////////////////////////////////////////////////
00016 // Import section
00017 // ////////////////////////////////////////////////////////////////////
00018 // STL
00019 #include <string>
00020 // StdAir
00021 #include <stdair/stdair_inventory_types.hpp>
00022 #include <stdair/stdair_service_types.hpp>
00023 #include <stdair/stdair_file.hpp>
00024 #include <stdair/basic/BasLogParams.hpp>
00025 #include <stdair/basic/BasDBParams.hpp>
00026 #include <stdair/basic/ServiceInitialisationType.hpp>
00027 #include <stdair/bom/TravelSolutionTypes.hpp>
00028 #include <stdair/bom/ConfigHolderStruct.hpp>
00029 #include <stdair/service/STDAIR_ServiceContext.hpp>
00030
00031 namespace stdair {
00032
00034   class BomRoot;
00035   struct EventStruct;
00036   struct ProgressStatusSet;
00037   struct BookingRequestStruct;
00038   class JSONString;
00039
00040
00044   class STDAIR_Service {
00045   public:
00046     // ////////// Constructors and destructors //////////
00050     STDAIR_Service();
00051
00063     STDAIR_Service (const BasLogParams&);
00064
00080     STDAIR_Service (const BasLogParams&, const BasDBParams&);
00081
00085     ~STDAIR_Service();
00086
00087
00088   public:
00089     // ////////////////// Business support methods /////////////////
00109     void buildSampleBom();
00110
00132     void buildDummyInventory (const CabinCapacity_T& iCabinCapacity);
00133
00146     void buildDummyLegSegmentAccesses (BomRoot&);
00147
00162     void buildSampleTravelSolutionForPricing (TravelSolutionList_T&);
00163
00181     void buildSampleTravelSolutions (TravelSolutionList_T&);
00182
00211     BookingRequestStruct buildSampleBookingRequest (const bool isForCRS = false);

00212
00216     void clonePersistentBom ();
00217
00218   public:
00219
00220     // ///////////////// Export support methods /////////////////
00233     std::string jsonExportFlightDateList (const AirlineCode_T& iAirlineCode = "al
      l",
00234                                           const FlightNumber_T& iFlightNumber = 0
      ) const;
```

```
00235
00246     std::string jsonExportFlightDateObjects (const AirlineCode_T&,
00247                                              const FlightNumber_T&,
00248                                              const Date_T& iDepartureDate) const;

00249
00256     std::string jsonExportEventObject (const EventStruct&) const;
00257
00264     std::string jsonExportConfiguration () const;
00265
00266   public:
00267
00268     // ################ Import support methods #################
00276     bool jsonImportConfiguration (const JSONString&) const;
00277
00278   public:
00279     // ################ Display support methods #################
00293     std::string list (const AirlineCode_T& iAirlineCode = "all",
00294                       const FlightNumber_T& iFlightNumber = 0) const;
00295
00302     std::string listAirportPairDateRange () const;
00303
00313     bool check (const AirlineCode_T&, const FlightNumber_T&,
00314                 const Date_T& iDepartureDate) const;
00315
00328     bool check (const AirportCode_T&, const AirportCode_T&,
00329                 const Date_T& iDepartureDate) const;
00330
00337     std::string configDisplay () const;
00338
00346     std::string csvDisplay () const;
00347
00356     std::string csvDisplay (const BomRoot&) const;
00357
00367     std::string csvDisplay (const AirlineCode_T&, const FlightNumber_T&,
00368                             const Date_T& iDepartureDate) const;
00369
00377     std::string csvDisplay (const TravelSolutionList_T&) const;
00378
00389     std::string csvDisplay (const AirportCode_T&, const AirportCode_T&,
00390                             const Date_T& iDepartureDate) const;
00391
00392
00393   public:
00394     // ################ Getters ##################
00403     BomRoot& getBomRoot() const;
00404
00413     BomRoot& getPersistentBomRoot() const;
00414
00420     BasLogParams getLogParams() const;
00421
00428     const BasDBParams& getDBParams() const;
00429
00438     const ServiceInitialisationType& getServiceInitialisationType() const;
00439
00440
00441   private:
00442     // ###### Construction and Destruction helper methods ######
00449     STDAIR_Service (const STDAIR_Service&);
00450
00455     void initServiceContext();
00456
00474     void logInit (const BasLogParams&);
00475
00481     void dbInit (const BasDBParams&);
00482
00498     void init();
```

```
00499
00503     void finalise();
00504
00505   public:
00506
00512     void importINIConfig (const ConfigINIFile&);
00513
00522     void importConfigValue (const std::string& iValue,
00523                             const std::string& iPath);
00524
00533     template <typename ValueType>
00534     bool exportConfigValue (ValueType& ioValue, const std::string& iPath);
00535
00540     void updateAirlineFeatures ();
00541
00542  private:
00543     // ///////// Service Context /////////
00547     STDAIR_ServiceContext* _stdairServiceContext;
00548   };
00549
00550   // ////////////////////////////////////////////////////////////////////
00551   template <typename ValueType>
00552   bool STDAIR_Service::exportConfigValue (ValueType& ioValue,
00553                                           const std::string& iPath) {
00554
00555     // Retrieve the StdAir service context
00556     assert (_stdairServiceContext != NULL);
00557     const STDAIR_ServiceContext& lSTDAIR_ServiceContext =
00558       *_stdairServiceContext;
00559
00560     // Retrieve the BOM tree root
00561     const ConfigHolderStruct& lConfigHolder =
00562       lSTDAIR_ServiceContext.getConfigHolder();
00563
00564     // Call the dedicated configuration holder method.
00565     return lConfigHolder.exportValue <ValueType> (ioValue, iPath);
00566   }
00567   // ////////////////////////////////////////////////////////////////////
00568
00569 }
00570 #endif // __STDAIR_STDAIR_HPP
```

33.631 stdair/stdair_service_types.hpp File Reference

```
#include <boost/shared_ptr.hpp>
```

Namespaces

- namespace stdair

  Handle on the StdAir library context.

Typedefs

- typedef boost::shared_ptr< STDAIR_Service > stdair::STDAIR_ServicePtr_T

33.632 stdair/stdair_service_types.hpp

```
#ifndef __STDAIR_STDAIR_SERVICE_HPP
#define __STDAIR_STDAIR_SERVICE_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// Boost (Extended STL)
#include <boost/shared_ptr.hpp>

namespace stdair {

  // Forward declarations
  class STDAIR_Service;

  typedef boost::shared_ptr<STDAIR_Service> STDAIR_ServicePtr_T;

}
#endif // __STDAIR_STDAIR_SERVICE_HPP
```

33.633 stdair/stdair_types.hpp File Reference

```
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_log.hpp>
#include <stdair/stdair_db.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_fare_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_service_types.hpp>
```

33.634 stdair/stdair_types.hpp

```
#ifndef __STDAIR_STDAIR_TYPES_HPP
#define __STDAIR_STDAIR_TYPES_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// StdAir
#include <stdair/stdair_exceptions.hpp>
#include <stdair/stdair_log.hpp>
#include <stdair/stdair_db.hpp>
#include <stdair/stdair_basic_types.hpp>
#include <stdair/stdair_demand_types.hpp>
#include <stdair/stdair_maths_types.hpp>
#include <stdair/stdair_fare_types.hpp>
#include <stdair/stdair_inventory_types.hpp>
#include <stdair/stdair_rm_types.hpp>
#include <stdair/stdair_date_time_types.hpp>
#include <stdair/stdair_service_types.hpp>

#endif // __STDAIR_STDAIR_TYPES_HPP
```

33.635 stdair/ui/cmdline/readline_autocomp.hpp File Reference

```
#include <string>
#include <iosfwd>
#include <cstdio>
#include <sys/types.h>
#include <sys/file.h>
#include <sys/stat.h>
#include <sys/errno.h>
#include <readline/readline.h>
#include <readline/history.h>
```

Classes

- struct COMMAND

Typedefs

- typedef int(∗ pt2Func)(char ∗)

Functions

- char ∗ getwd ()
- char ∗ xmalloc (size_t)
- int com_list (char ∗)
- int com_view (char ∗)
- int com_rename (char ∗)
- int com_stat (char ∗)
- int com_pwd (char ∗)
- int com_delete (char ∗)
- int com_help (char ∗)
- int com_cd (char ∗)
- int com_quit (char ∗)
- char ∗ stripwhite (char ∗iString)
- COMMAND ∗ find_command (char ∗iString)
- char ∗ dupstr (char ∗iString)
- int execute_line (char ∗line)
- char ∗ command_generator (char ∗text, int state)
- char ∗∗ fileman_completion (char ∗text, int start, int end)
- void initialize_readline ()
- void too_dangerous (char ∗caller)
- int valid_argument (char ∗caller, char ∗arg)

Variables

- COMMAND commands [ ]
- int done
- static char syscom [1024]

33.635.1 Typedef Documentation

33.635.1.1 typedef int(∗ pt2Func)(char ∗)

Definition at line 35 of file readline_autocomp.hpp.

33.635.2 Function Documentation

33.635.2.1 char∗ getwd ()

readline_autocomp.hpp -- A tiny application which demonstrates how to use the GNU Readline library. This application interactively allows users to manipulate files and their modes.

Referenced by com_pwd().

33.635.2.2 char∗ xmalloc (size_t)

Referenced by dupstr().

33.635.2.3 void com_list (char ∗ *arg*)

List the file(s) named in arg.

Definition at line 264 of file readline_autocomp.hpp.

33.635.2.4 int com_view (char ∗ *arg*)

Definition at line 274 of file readline_autocomp.hpp.

References valid_argument().

33.635.2.5 int com_rename (char ∗ *arg*)

Definition at line 284 of file readline_autocomp.hpp.

References too_dangerous().

33.635.2.6 int com_stat (char ∗ *arg*)

Definition at line 289 of file readline_autocomp.hpp.

References valid_argument().

33.635.2.7 int com_pwd (char * *ignore*)

Definition at line 367 of file readline_autocomp.hpp.

References getwd().

Referenced by com_cd().

33.635.2.8 int com_delete (char * *arg*)

Definition at line 315 of file readline_autocomp.hpp.

References too_dangerous().

33.635.2.9 int com_help (char * *arg*)

Print out help for ARG, or for all of the commands if ARG is not present.

Definition at line 324 of file readline_autocomp.hpp.

References COMMAND::name.

33.635.2.10 int com_cd (char * *arg*)

Definition at line 356 of file readline_autocomp.hpp.

References com_pwd().

33.635.2.11 int com_quit (char * *arg*)

Definition at line 381 of file readline_autocomp.hpp.

33.635.2.12 char * stripwhite (char * *string*)

Strip whitespace from the start and end of STRING. Return a pointer into STRING.

Definition at line 152 of file readline_autocomp.hpp.

33.635.2.13 COMMAND * find_command (char * *name*)

Look up NAME as the name of a command, and return a pointer to that command. Return a NULL pointer if NAME isn't a command name.

Definition at line 136 of file readline_autocomp.hpp.

References COMMAND::name.

Referenced by execute_line().

33.635.2.14 char* dupstr (char * *iString*)

Duplicate a string

Definition at line 85 of file readline_autocomp.hpp.

References xmalloc().

Referenced by command_generator().

33.635.2.15 int execute_line (char * *line*)

Execute a command line.

Definition at line 94 of file readline_autocomp.hpp.

References find_command(), and COMMAND::func.

33.635.2.16 char * command_generator (char * *text*, int *state*)

Generator function for command completion. STATE lets us know whether to start from scratch; without any state (i.e. STATE == 0), then we start at the top of the list.

Definition at line 222 of file readline_autocomp.hpp.

References dupstr().

Referenced by fileman_completion().

33.635.2.17 char ** fileman_completion (char * *text*, int *start*, int *end*)

Attempt to complete on the contents of TEXT. START and END bound the region of rl_line_buffer that contains the word to complete. TEXT is the word to complete. We can use the entire contents of rl_line_buffer in case we want to do some simple parsing. Return the array of matches, or NULL if there aren't any.

Definition at line 200 of file readline_autocomp.hpp.

References command_generator().

Referenced by initialize_readline().

33.635.2.18 void initialize_readline ()

Tell the GNU Readline library how to complete. We want to try to complete on command names if this is the first word in the line, or on filenames if not.

Definition at line 185 of file readline_autocomp.hpp.

References fileman_completion().

33.635.2.19 void too_dangerous (char * *caller*)

Definition at line 387 of file readline_autocomp.hpp.

Referenced by com_delete(), and com_rename().

33.635.2.20 int valid_argument (char * *caller*, char * *arg*)

Definition at line 395 of file readline_autocomp.hpp.

Referenced by com_stat(), and com_view().

33.635.3 Variable Documentation

33.635.3.1 COMMAND commands[ ]

Initial value:

```
{
 { "cd", (*com_cd)(), "Change to directory DIR" },
 { "delete", com_delete, "Delete FILE" },
 { "help", com_help, "Display this text" },
 { "?", com_help, "Synonym for `help'" },
 { "list", com_list, "List files in DIR" },
 { "ls", com_list, "Synonym for `list'" },
 { "pwd", com_pwd, "Print the current working directory" },
 { "quit", com_quit, "Quit using airinv" },
 { "rename", com_rename, "Rename FILE to NEWNAME" },
 { "stat", com_stat, "Print out statistics on FILE" },
 { "view", com_view, "View the contents of FILE" },
 { (char*) NULL, (pt2Func) NULL, (char*) NULL }
}
```

Definition at line 58 of file readline_autocomp.hpp.

33.635.3.2 int done

When non-zero, this global means the user is done using this program.

Definition at line 80 of file readline_autocomp.hpp.

33.635.3.3 char syscom[1024] [static]

String to pass to system(). This is for the LIST, VIEW and RENAME commands.

Definition at line 259 of file readline_autocomp.hpp.

33.636 stdair/ui/cmdline/readline_autocomp.hpp

```
00001
00006 #ifndef __AIRINV_READLINE_AUTOCOMP_HPP
00007 #define __AIRINV_READLINE_AUTOCOMP_HPP
00008
00009 // STL
00010 #include <string>
00011 #include <iosfwd>
00012 #include <cstdio>
00013 #include <sys/types.h>
00014 #include <sys/file.h>
00015 #include <sys/stat.h>
00016 #include <sys/errno.h>
00017
00018 #include <readline/readline.h>
00019 #include <readline/history.h>
00020
00021 extern char* getwd();
00022 extern char* xmalloc (size_t);
00023
00024 /* The names of functions that actually do the manipulation. */
00025 int com_list (char*);
00026 int com_view (char*);
00027 int com_rename (char*);
00028 int com_stat (char*);
00029 int com_pwd (char*);
00030 int com_delete (char*);
00031 int com_help (char*);
00032 int com_cd (char*);
00033 int com_quit (char*);
00034
00035 typedef int (*pt2Func) (char*);
00036
00041 typedef struct {
00045   char const* name;
00046
00050   pt2Func *func;
00051
00055   char *doc;
00056 } COMMAND;
00057
00058 COMMAND commands[] = {
00059   { "cd", (*com_cd)(), "Change to directory DIR" },
00060   { "delete", com_delete, "Delete FILE" },
00061   { "help", com_help, "Display this text" },
00062   { "?", com_help, "Synonym for `help'" },
00063   { "list", com_list, "List files in DIR" },
00064   { "ls", com_list, "Synonym for `list'" },
00065   { "pwd", com_pwd, "Print the current working directory" },
00066   { "quit", com_quit, "Quit using airinv" },
00067   { "rename", com_rename, "Rename FILE to NEWNAME" },
00068   { "stat", com_stat, "Print out statistics on FILE" },
00069   { "view", com_view, "View the contents of FILE" },
00070   { (char*) NULL, (pt2Func) NULL, (char*) NULL }
00071 };
00072
00073 // Forward declarations
00074 char* stripwhite (char* iString);
00075 COMMAND* find_command (char* iString);
00076
00080 int done;
00081
00085 char* dupstr (char* iString) {
00086   char* r = xmalloc (std::strlen (iString) + 1);
00087   strcpy (r, iString);
00088   return r;
```

```
}

int execute_line (char* line) {
  register int i;
  COMMAND* command;
  char* word;

  /* Isolate the command word. */
  i = 0;
  while (line[i] && whitespace (line[i])) {
    i++;
  }
  word = line + i;

  while (line[i] && !whitespace (line[i])) {
    i++;
  }

  if (line[i]) {
    line[i++] = '\0';
  }

  command = find_command (word);

  if (!command) {
    std::cerr << word << ": No such command for airinv." << std::endl;
    return -1;
  }

  /* Get argument to command, if any. */
  while (whitespace (line[i])) {
    i++;
  }

  word = line + i;

  /* Call the function. */
  return (*(command->func)) (word);
}

COMMAND* find_command (char* name) {
  register int i;

  for (i = 0; commands[i].name; i++) {
    if (strcmp (name, commands[i].name) == 0) {
      return (&commands[i]);
    }
  }

  return (COMMAND*) NULL;
}

char* stripwhite (char* string) {
  register char *s, *t;

  for (s = string; whitespace (*s); s++) {
  }

  if (*s == 0) {
    return s;
  }

  t = s + strlen (s) - 1;
  while (t > s && whitespace (*t)) {
    t--;
  }
  *++t = '\0';
```

```
00167
00168   return s;
00169 }
00170
00171 /* ************************************************************** */
00172 /*                                                                */
00173 /*                  Interface to Readline Completion              */
00174 /*                                                                */
00175 /* ************************************************************** */
00176
00177 char* command_generator (char* text, int state);
00178 char** fileman_completion (char* text, int start, int end);
00179
00185 void initialize_readline() {
00186   /* Allow conditional parsing of the ~/.inputrc file. */
00187   rl_readline_name = "airinv";
00188
00189   /* Tell the completer that we want a crack first. */
00190   rl_attempted_completion_function = (rl_completion_func_t*) fileman_completion;
00191 }
00192
00200 char** fileman_completion (char* text, int start, int end) {
00201   char **matches;
00202
00203   matches = (char**) NULL;
00204
00210   if (start == 0) {
00211     matches = completion_matches (text, command_generator);
00212   }
00213
00214   return matches;
00215 }
00216
00222 char* command_generator (char* text, int state) {
00223   static int list_index, len;
00224   char* name;
00225
00231   if (!state) {
00232     list_index = 0;
00233     len = strlen (text);
00234   }
00235
00236   /* Return the next name which partially matches from the command list. */
00237   while (name = commands[list_index].name) {
00238     ++list_index;
00239
00240     if (strncmp (name, text, len) == 0) {
00241       return dupstr (name);
00242     }
00243   }
00244
00245   /* If no names matched, then return NULL. */
00246   return (char*) NULL;
00247 }
00248
00249 /* ************************************************************** */
00250 /*                                                                */
00251 /*                        airinv Commands                         */
00252 /*                                                                */
00253 /* ************************************************************** */
00254
00259 static char syscom[1024];
00260
00264 void com_list (char* arg) {
00265   if (!arg) {
00266     arg = "";
00267   }
```

```
00268
00269   std::ostringstream oStr;
00270   oStr << "ls -FClg " << arg;
00271   return system (oStr.c_str());
00272 }
00273
00274 int com_view (char* arg) {
00275   if (!valid_argument ("view", arg)) {
00276     return 1;
00277   }
00278
00279   std::ostringstream oStr;
00280   oStr << "more " << arg;
00281   return system (syscom);
00282 }
00283
00284 int com_rename (char* arg) {
00285   too_dangerous ("rename");
00286   return 1;
00287 }
00288
00289 int com_stat (char* arg) {
00290   struct stat finfo;
00291
00292   if (!valid_argument ("stat", arg)) {
00293     return 1;
00294   }
00295
00296   if (stat (arg, &finfo) == -1) {
00297     perror (arg);
00298     return 1;
00299   }
00300
00301   std::cout << "Statistics for `" << arg << "':" << std::endl;
00302
00303   const std::string lPluralEnd1 = (finfo.st_nlink == 1) ? "" : "s";
00304   const std::string lPluralEnd2 = (finfo.st_size == 1) ? "" : "s";
00305   std::cout << arg << " has "
00306             << finfo.st_nlink << " link" << lPluralEnd1 << ", and is "
00307             << finfo.st_size << " byte" << lPluralEnd2 << " in length."
00308             << std::endl;
00309   std::cout << " Inode Last Change at: " << ctime (&finfo.st_ctime) << std::endl;

00310   std::cout << " Last access at: " << ctime (&finfo.st_atime) << std::endl;
00311   std::cout << " Last modified at: " << ctime (&finfo.st_mtime) << std::endl;
00312   return 0;
00313 }
00314
00315 int com_delete (char* arg) {
00316   too_dangerous ("delete");
00317   return 1;
00318 }
00319
00324 int com_help (char* arg) {
00325   register int i;
00326   int printed = 0;
00327
00328   for (i = 0; commands[i].name; i++) {
00329     if (!*arg || (strcmp (arg, commands[i].name) == 0)) {
00330       printf ("%s\t\t%s.\n", commands[i].name, commands[i].doc);
00331       printed++;
00332     }
00333   }
00334
00335   if (!printed) {
00336     printf ("No commands match `%s'.  Possibilties are:\n", arg);
00337
```

```
    for (i = 0; commands[i].name; i++) {
      /* Print in six columns. */
      if (printed == 6) {
        printed = 0;
        printf ("\n");
      }

      printf ("%s\t", commands[i].name);
      printed++;
    }

    if (printed)
      printf ("\n");
  }
  return 0;
}

/* Change to the directory ARG. */
int com_cd (char* arg) {
  if (chdir (arg) == -1) {
    perror (arg);
    return 1;
  }

  com_pwd ("");
  return 0;
}

/* Print out the current working directory. */
int com_pwd (char* ignore) {
  char dir[1024], *s;

  s = getwd (dir);
  if (s == 0) {
    printf ("Error getting pwd: %s\n", dir);
    return 1;
  }

  printf ("Current directory is %s\n", dir);
  return 0;
}

/* The user wishes to quit using this program.  Just set DONE non-zero. */
int com_quit (char* arg) {
  done = 1;
  return 0;
}

/* Function which tells you that you can't do this. */
void too_dangerous (char* caller) {
  fprintf (stderr,
           "%s: Too dangerous for me to distribute.  Write it yourself.\n",
           caller);
}

/* Return non-zero if ARG is a valid argument for CALLER, else print
 *    an error message and return zero. */
int valid_argument (char* caller, char* arg) {
  if (!arg || !*arg) {
    fprintf (stderr, "%s: Argument required.\n", caller);
    return 0;
  }

  return 1;
}

#endif // _AIRINV_READLINE_AUTOCOMP_HPP
```

33.637 stdair/ui/cmdline/SReadline.hpp File Reference

C++ wrapper around libreadline. `#include <stdio.h>`

```
#include <readline/readline.h>
#include <readline/history.h>
#include <readline/keymaps.h>
#include <string>
#include <fstream>
#include <vector>
#include <stdexcept>
#include <map>
#include <boost/algorithm/string/trim.hpp>
#include <boost/tokenizer.hpp>
#include <boost/function.hpp>
```

Classes

- class swift::SKeymap

  The readline keymap wrapper.

- class swift::SReadline

  The readline library wrapper.

Namespaces

- namespace swift

  The wrapper namespace.

33.637.1 Detailed Description

C++ wrapper around libreadline. Supported: editing, history, custom completers, keymaps. Attention: implementation is not thread safe! It is mainly because the readline library provides pure C interface and has many calls for an "atomic" completion operation

Definition in file SReadline.hpp.

33.638 stdair/ui/cmdline/SReadline.hpp

```
00001
00011 //
00012 // Date:      17 December 2005
00013 //            03 April    2006
00014 //            20 April    2006
00015 //            07 May      2006
00016 //
00017 // Copyright (c) Sergey Satskiy 2005 - 2006
00018 //               <sergesatsky@yahoo.com>
00019 //
00020 // Permission to copy, use, modify, sell and distribute this software
00021 // is granted provided this copyright notice appears in all copies.
00022 // This software is provided "as is" without express or implied
00023 // warranty, and with no claim as to its suitability for any purpose.
00024 //
00025
00026 #ifndef SREADLINE_H
00027 #define SREADLINE_H
00028
00029 #include <stdio.h>
00030
00031 #include <readline/readline.h>
00032 #include <readline/history.h>
00033 #include <readline/keymaps.h>
00034
00035 #include <string>
00036 #include <fstream>
00037 #include <vector>
00038 #include <stdexcept>
00039 #include <map>
00040
00041 #include <boost/algorithm/string/trim.hpp>
00042 #include <boost/tokenizer.hpp>
00043 #include <boost/function.hpp>
00044
00045
00050 namespace {
00054   typedef std::vector<std::string> TokensStorage;
00055
00059   typedef std::vector<TokensStorage> CompletionsStorage;
00060
00064   typedef boost::function<int (int, int)> KeyCallback;
00065
00069   typedef std::map<int, KeyCallback> KeysBind;
00070
00074   const size_t DefaultHistoryLimit (64);
00075
00079   CompletionsStorage Completions;
00080
00084   TokensStorage Tokens;
00085
00089   std::map<Keymap, KeysBind> Keymaps;
00090
00094   bool KeymapWasSetup (false);
00095
00099   Keymap Earlykeymap (0);
00100
00101
00108   char* Generator (const char* text, int State);
00109
00110
00118   char** UserCompletion (const char* text, int start, int end);
00119
00120
00128   int KeyDispatcher (int Count, int Key);
```

```


int StartupHook (void);


template <typename  Container>
bool AreTokensEqual (const Container& Pattern, const Container& Input) {
  if (Input.size() > Pattern.size()) {
    return false;
  }

  typename Container::const_iterator k (Pattern.begin());
  typename Container::const_iterator j (Input.begin());
  for ( ; j != Input.end(); ++k, ++j) {
    const std::string lPattern = *k;
    if (lPattern == "%file") {
      continue;
    }

    const std::string lInput = *j;
    if (lPattern != lInput) {
      return false;
    }
  }
  return true;
}

// See description near the prototype
template <typename  ContainerType>
void SplitTokens (const std::string& Source, ContainerType& Container) {
  typedef boost::tokenizer<boost::char_separator<char> > TokenizerType;

  // Set of token separators
  boost::char_separator<char> Separators (" \t\n");
  // Tokens provider
  TokenizerType Tokenizer (Source, Separators);

  Container.clear();
  for (TokenizerType::const_iterator k (Tokenizer.begin());
       k != Tokenizer.end(); ++k) {
    // Temporary storage for the token, in order to trim that latter
    std::string SingleToken (*k);

    boost::algorithm::trim (SingleToken);
    Container.push_back (SingleToken);
  }
}

// See description near the prototype
char** UserCompletion (const char* text, int start, int end) {
  // No default completion at all
  rl_attempted_completion_over = 1;

  if (Completions.empty() == true) {
    return NULL;
  }

  // Memorise all the previous tokens
  std::string PreInput (rl_line_buffer, start);
  SplitTokens (PreInput, Tokens);

  // Detect whether we should call the standard file name completer
  // or a custom one
  bool FoundPretender (false);

  for (CompletionsStorage::const_iterator k (Completions.begin());
       k != Completions.end(); ++k) {
```

```
      const TokensStorage& lTokenStorage = *k;
      if (AreTokensEqual (lTokenStorage, Tokens) == false) {
        continue;
      }

      if (lTokenStorage.size() > Tokens.size()) {
        FoundPretender = true;
        if (lTokenStorage [Tokens.size()] == "%file") {
          // Standard file name completer - called for the "%file" keyword
          return rl_completion_matches (text, rl_filename_completion_function);
        }
      }
    }

    if (FoundPretender) {
      return rl_completion_matches (text, Generator);
    }
    return NULL;
  }

  // See description near the prototype
  char* Generator (const char* text, int State) {
    static int Length;
    static CompletionsStorage::const_iterator Iterator;

    if ( State == 0 ) {
      Iterator = Completions.begin();
      Length = strlen (text);
    }

    for ( ; Iterator != Completions.end(); ++Iterator) {
      const TokensStorage& lCompletion = *Iterator;
      if (AreTokensEqual (lCompletion, Tokens) == false) {
        continue;
      }

      if (lCompletion.size() > Tokens.size()) {
        if (lCompletion [Tokens.size()] == "%file") {
          continue;
        }

        const char* lCompletionCharStr (lCompletion [Tokens.size()].c_str());
        if (strncmp (text, lCompletionCharStr, Length) == 0) {
          // Readline will free the allocated memory
          const size_t lCompletionSize = strlen (lCompletionCharStr) + 1;
          char* NewString (static_cast<char*> (malloc (lCompletionSize)));
          strcpy (NewString, lCompletionCharStr);

          ++Iterator;

          return NewString;
        }
      }
    }

    return NULL;
  }


  // See the description near the prototype
  int KeyDispatcher  (int Count, int  Key ) {
    std::map< Keymap, KeysBind >::iterator Set (Keymaps.find (rl_get_keymap()));
    if (Set == Keymaps.end()) {
      // Most probably it happens bacause the header was
      // included into many compilation units and the
      // keymap setting calls were made in different files.
      // This is the problem of "global" data.
```

```
      // The storage of all the registered keymaps is in anonymous
      // namespace.
      throw std::runtime_error ("Error selecting a keymap.");
    }

    (Set->second)[Key] (Count, Key);
    return 0;
  }

  // See the description near the prototype
  int StartupHook (void) {
    if (KeymapWasSetup) {
      rl_set_keymap (Earlykeymap);
    }
    return 0;
  }

} // Anonymous namespace


namespace swift {

  class SKeymap {
  private:
    // Readline keymap
    Keymap keymap;

  public:
    explicit SKeymap (bool PrintableBound = false) : keymap (NULL) {
      if (PrintableBound == true) {
        // Printable characters are bound
        keymap = rl_make_keymap();

      } else {
        // Empty keymap
        keymap = rl_make_bare_keymap();
      }

      if (keymap == NULL) {
        throw std::runtime_error ("Cannot allocate keymap.");
      }

      // Register a new keymap in the global list
      Keymaps [keymap] = KeysBind();
    }

    explicit SKeymap (Keymap Pattern) : keymap (rl_copy_keymap (Pattern)) {
      if ( keymap == NULL ) {
        throw std::runtime_error( "Cannot allocate keymap." );
      }

      // Register a new keymap in the global list
      Keymaps [keymap] = KeysBind();
    }

    ~SKeymap() {
      // Deregister the keymap
      Keymaps.erase (keymap);
      rl_discard_keymap (keymap);
    }

    void Bind (int  Key, KeyCallback  Callback) {
      Keymaps [keymap][Key] = Callback;

      if (rl_bind_key_in_map (Key, KeyDispatcher, keymap) != 0) {
        // Remove from the map just bound key
        Keymaps [keymap].erase (Key);
```

```
00372         throw std::runtime_error ("Invalid key.");
00373       }
00374     }
00375
00381     void Unbind (int  Key) {
00382       rl_unbind_key_in_map (Key, keymap);
00383       Keymaps [keymap].erase (Key);
00384     }
00385
00386     // void Bind (const std::string& Sequence, boost::function<int (int, int)>);
00387     // void Unbind (std::string&  Sequence);
00388
00389   public:
00395     SKeymap (const SKeymap& rhs) {
00396       if (this == &rhs) {
00397         return;
00398       }
00399       keymap = rl_copy_keymap (rhs.keymap);
00400     }
00401
00407     SKeymap& operator= (const SKeymap& rhs) {
00408       if (this == &rhs) {
00409         return *this;
00410       }
00411       keymap = rl_copy_keymap (rhs.keymap);
00412       return *this;
00413     }
00414
00415     friend class SReadline;
00416   };
00417
00424   class SReadline {
00425   public:
00431     SReadline (const size_t Limit = DefaultHistoryLimit) :
00432       HistoryLimit (Limit), HistoryFileName (""),
00433       OriginalCompletion (rl_attempted_completion_function) {
00434       rl_startup_hook = StartupHook;
00435       rl_attempted_completion_function = UserCompletion;
00436       using_history();
00437     }
00438
00446     SReadline( const std::string &  historyFileName,
00447                const size_t  Limit = DefaultHistoryLimit ) :
00448       HistoryLimit( Limit ),
00449       HistoryFileName( historyFileName ),
00450       OriginalCompletion( rl_attempted_completion_function )
00451     {
00452       rl_startup_hook = StartupHook;
00453       rl_attempted_completion_function = UserCompletion;
00454       using_history();
00455       LoadHistory( HistoryFileName );
00456     }
00457
00462     ~SReadline() {
00463       rl_attempted_completion_function = OriginalCompletion;
00464       SaveHistory (HistoryFileName);
00465     }
00466
00473     std::string GetLine (const std::string& Prompt) {
00474       bool Unused;
00475       return GetLine (Prompt, Unused);
00476     }
00477
00486     template <typename  Container>
00487     std::string GetLine (const std::string& Prompt, Container& ReadTokens) {
00488       bool Unused;
00489       return GetLine (Prompt, ReadTokens, Unused);
```

```
00490     }
00491
00501     template <typename  Container>
00502     std::string GetLine (const std::string& Prompt, Container& ReadTokens,
00503                          bool& BreakOut) {
00504       std::string Input (GetLine (Prompt, BreakOut));
00505       SplitTokens (Input, ReadTokens);
00506       return Input;
00507     }
00508
00509
00517     std::string GetLine (const std::string& Prompt, bool& BreakOut) {
00518       BreakOut = true;
00519
00520       char* ReadLine (readline (Prompt.c_str()));
00521       if (ReadLine == NULL) {
00522         return std::string();
00523       }
00524
00525       // It's OK
00526       BreakOut = false;
00527       std::string Input (ReadLine);
00528       free (ReadLine); ReadLine = NULL;
00529
00530       boost::algorithm::trim (Input);
00531       if (Input.empty() == false) {
00532         if (history_length == 0
00533             || Input != history_list()[ history_length - 1 ]->line) {
00534           add_history (Input.c_str());
00535
00536           if (history_length >= static_cast<int> (HistoryLimit)) {
00537             stifle_history (HistoryLimit);
00538           }
00539         }
00540       }
00541
00542       return Input;
00543     }
00544
00545
00551     template <typename ContainerType>
00552     void GetHistory (ContainerType&  Container) {
00553       for (int k (0); k < history_length; ++k ) {
00554         Container.push_back (history_list()[k]->line);
00555       }
00556     }
00557
00564     bool SaveHistory (std::ostream& OS) {
00565       if (!OS) {
00566         return false;
00567       }
00568
00569       for (int k (0); k < history_length; ++k) {
00570         OS << history_list()[ k ]->line << std::endl;
00571       }
00572       return true;
00573     }
00574
00581     bool SaveHistory (const std::string&  FileName) {
00582       if (FileName.empty() == true) {
00583         return false;
00584       }
00585
00586       std::ofstream OS (FileName.c_str());
00587       return SaveHistory (OS);
00588     }
00589
```

```
    void ClearHistory() {
      clear_history();
    }

    bool LoadHistory (std::istream& IS) {
      if (!IS) {
        return false;
      }

      ClearHistory();
      std::string OneLine;

      while (!getline (IS, OneLine).eof()) {
        boost::algorithm::trim( OneLine );
        if ((history_length == 0)
            || OneLine != history_list()[history_length - 1]->line) {
          add_history (OneLine.c_str());
        }
      }
      stifle_history (HistoryLimit);
      return true;
    }

    bool LoadHistory (const std::string&  FileName) {
      if (FileName.empty() == true) {
        return false;
      }

      std::ifstream IS (FileName.c_str());
      return LoadHistory (IS);
    }

    template <typename  ContainerType>
    void RegisterCompletions (const ContainerType& Container) {
      Completions.clear();
      for (typename ContainerType::const_iterator k (Container.begin());
           k != Container.end(); ++k) {
        std::vector<std::string> OneLine;
        const std::string& kStr = static_cast<std::string> (*k);

        SplitTokens (kStr, OneLine);
        Completions.push_back (OneLine);
      }
    }

    void SetKeymap (SKeymap& NewKeymap) {
      rl_set_keymap (NewKeymap.keymap);
      KeymapWasSetup = true;
      Earlykeymap = NewKeymap.keymap;
    }


  private:
    // ///////////////////////////// Attributes /////////////////////////
    const size_t HistoryLimit;

    const std::string HistoryFileName;

    rl_completion_func_t* OriginalCompletion;
  };

}; // namespace swift

#endif
```

33.639 test/stdair/MPBomRoot.cpp File Reference

33.640 test/stdair/MPBomRoot.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <cassert>
00010 // StdAir Test
00011 #include <test/stdair/MPBomRoot.hpp>
00012
00013 namespace myprovider {
00014
00015   // ////////////////////////////////////////////////////////////////////
00016   BomRoot::BomRoot (const Key_T& iKey) : stdair::BomRoot (iKey) {
00017   }
00018
00019   // ////////////////////////////////////////////////////////////////////
00020   BomRoot::~BomRoot () {
00021   }
00022
00023 }
```

33.641 test/stdair/MPBomRoot.hpp File Reference

33.642 test/stdair/MPBomRoot.hpp

```
00001 #ifndef __MYPROVIDER_BOMROOT_HPP
00002 #define __MYPROVIDER_BOMROOT_HPP
00003
00008 // ////////////////////////////////////////////////////////////////////
00009 // Import section
00010 // ////////////////////////////////////////////////////////////////////
00011 // STL
00012 #include <string>
00013 // StdAir
00014 #include <stdair/bom/BomRoot.hpp>
00015
00016 namespace myprovider {
00017
00020   class BomRoot : public stdair::BomRoot {
00021   public:
00022     // /////////// Display support methods /////////
00024     std::string toString() const { return describeKey(); }
00025
00028     const std::string describeKey() const { return std::string (""); }
00029
00030   public:
00034     BomRoot (const Key_T&);
00036     ~BomRoot();
00038     BomRoot ();
00039     BomRoot (const BomRoot&);
00040   };
00041
00042 }
00046 #endif // __MYPROVIDER_BOMROOT_HPP
```

33.643 test/stdair/MPInventory.cpp File Reference

33.644 test/stdair/MPInventory.cpp

```

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <cassert>
// StdAir
#include <stdair/stdair_inventory_types.hpp>
// StdAir Test
#include <test/stdair/MPInventory.hpp>

namespace myprovider {

  // ////////////////////////////////////////////////////////////////////
  Inventory::Inventory (const Key_T& iKey) : stdair::Inventory (iKey) {
  }

  // ////////////////////////////////////////////////////////////////////
  Inventory::~Inventory () {
  }

  // ////////////////////////////////////////////////////////////////////
  std::string Inventory::toString() const {
    std::ostringstream oStr;
    oStr << _key.toString();
    return oStr.str();
  }

  // ////////////////////////////////////////////////////////////////////
  const std::string Inventory::describeKey() const {
    return _key.toString();
  }

}
```

33.645 test/stdair/MPInventory.hpp File Reference

33.646 test/stdair/MPInventory.hpp

```
#ifndef __MYPROVIDER_INVENTORY_HPP
#define __MYPROVIDER_INVENTORY_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
// STL
#include <list>
// StdAir
#include <stdair/bom/Inventory.hpp>

namespace myprovider {

  class Inventory : public stdair::Inventory {
  public:
    // /////////// Display support methods /////////
    std::string toString() const;

    const std::string describeKey() const;

  public:
    Inventory (const Key_T&);
    ~Inventory();
    Inventory ();
    Inventory (const Inventory&);
  };

  // ////////// Type definitions /////////
  typedef std::list<Inventory*> InventoryList_T;

}
#endif // __MYPROVIDER_INVENTORY_HPP
```

33.647 test/stdair/StandardAirlineITTestSuite.cpp File Reference

33.648 test/stdair/StandardAirlineITTestSuite.cpp

```
00001
00005 // ////////////////////////////////////////////////////////////////////
00006 // Import section
00007 // ////////////////////////////////////////////////////////////////////
00008 // STL
00009 #include <sstream>
00010 #include <fstream>
00011 #include <string>
00012 // Boost MPL
00013 #include <boost/mpl/push_back.hpp>
00014 #include <boost/mpl/vector.hpp>
00015 #include <boost/mpl/at.hpp>
00016 #include <boost/mpl/assert.hpp>
00017 #include <boost/type_traits/is_same.hpp>
00018 // Boost Unit Test Framework (UTF)
00019 #define BOOST_TEST_DYN_LINK
00020 #define BOOST_TEST_MAIN
00021 #define BOOST_TEST_MODULE StdAirTest
00022 #if BOOST_VERSION >= 103900
00023 #include <boost/test/unit_test.hpp>
00024 #else  // BOOST_VERSION >= 103900
00025 #include <boost/test/test_tools.hpp>
00026 #include <boost/test/results_reporter.hpp>
00027 #include <boost/test/unit_test_suite.hpp>
00028 #include <boost/test/output_test_stream.hpp>
00029 #include <boost/test/unit_test_log.hpp>
00030 #include <boost/test/framework.hpp>
00031 #include <boost/test/detail/unit_test_parameters.hpp>
00032 #endif // BOOST_VERSION >= 103900
00033 // Boost Serialisation
00034 #include <boost/archive/text_oarchive.hpp>
00035 #include <boost/archive/text_iarchive.hpp>
00036 // StdAir
00037 #include <stdair/stdair_inventory_types.hpp>
00038 #include <stdair/service/Logger.hpp>
00039 #include <stdair/STDAIR_Service.hpp>
00040 #include <stdair/basic/float_utils.hpp>
00041 #include <stdair/bom/BomDisplay.hpp>
00042 #include <stdair/bom/BomRoot.hpp>
00043 #include <stdair/bom/BomManager.hpp>
00044 #include <stdair/factory/FacBom.hpp>
00045 #include <stdair/factory/FacBomManager.hpp>
00046 // StdAir Test Suite
00047 #include <test/stdair/StdairTestLib.hpp>
00048 #include <test/stdair/MPInventory.hpp>
00049
00050 namespace boost_utf = boost::unit_test;
00051
00052 #if BOOST_VERSION >= 103900
00053
00054 // (Boost) Unit Test XML Report
00055 std::ofstream utfReportStream ("StandardAirlineITTestSuite_utfresults.xml");
00056
00060 struct UnitTestConfig {
00062   UnitTestConfig() {
00063     boost_utf::unit_test_log.set_stream (utfReportStream);
00064     boost_utf::unit_test_log.set_format (boost_utf::XML);
00065     boost_utf::unit_test_log.set_threshold_level (boost_utf::log_test_units);
00066     // boost_utf::unit_test_log.set_threshold_level (boost_utf::log_successful_te
      sts);
00067   }
00068
00070   ~UnitTestConfig() {
00071   }
00072 };
```

```


// /////////////// Main: Unit Test Suite //////////////

// Set the UTF configuration (re-direct the output to a specific file)
BOOST_GLOBAL_FIXTURE (UnitTestConfig);

// Start the test suite
BOOST_AUTO_TEST_SUITE (master_test_suite)


BOOST_AUTO_TEST_CASE (float_comparison_test) {
  float a = 0.2f;
  a = 5*a;
  const float b = 1.0f;

  // Test the Boost way
  BOOST_CHECK_MESSAGE (a == b, "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
  BOOST_CHECK_CLOSE (a, b, 0.0001);

  // Test the Google way
  const FloatingPoint<float> lhs (a), rhs (b);
  BOOST_CHECK_MESSAGE (lhs.AlmostEquals (rhs),
                       "The two floats (" << a << " and " << b
                       << ") should be equal, but are not");
}

BOOST_AUTO_TEST_CASE (mpl_structure_test) {
  const stdair::ClassCode_T lBookingClassCodeA ("A");
  const stdair_test::BookingClass lA (lBookingClassCodeA);
  const stdair_test::Cabin lCabin (lA);

  BOOST_CHECK_EQUAL (lCabin.toString(), lBookingClassCodeA);
  BOOST_CHECK_MESSAGE (lCabin.toString() == lBookingClassCodeA,
                       "The cabin key, '" << lCabin.toString()
                       << "' is not equal to '" << lBookingClassCodeA << "'");

  // MPL
  typedef boost::mpl::vector<stdair_test::BookingClass> MPL_BookingClass;
  typedef boost::mpl::push_back<MPL_BookingClass,
                                stdair_test::Cabin>::type types;

  if (boost::is_same<stdair_test::BookingClass,
                     stdair_test::Cabin::child>::value == false) {
    BOOST_ERROR ("The two types mut be equal, but are not");
  }

  if (boost::is_same<boost::mpl::at_c<types, 1>::type,
                     stdair_test::Cabin>::value == false) {
    BOOST_ERROR ("The type must be stdair_test::Cabin, but is not");
  }
}

BOOST_AUTO_TEST_CASE (stdair_service_initialisation_test) {
  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_init.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
```

```
  stdair::STDAIR_Service stdairService (lLogParams);

  // Retrieve (a reference on) the top of the persistent BOM tree
  stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lPersistentBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
                       << "', should be equal to '" << lBomRootString
                       << "', but is not.");

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // DEBUG: Display the whole BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_instantiation_test) {
  // Step 0.0: initialisation
  // Create the root of a Bom tree (i.e., a BomRoot object)
  stdair::BomRoot& lBomRoot =
    stdair::FacBom<stdair::BomRoot>::instance().create();

  // Step 0.1: Inventory level
  // Create an Inventory (BA)
  const stdair::AirlineCode_T lBAAirlineCode ("BA");
  const stdair::InventoryKey lBAKey (lBAAirlineCode);
  myprovider::Inventory& lBAInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lBAKey);
  stdair::FacBomManager::addToList (lBomRoot, lBAInv);

  BOOST_CHECK_EQUAL (lBAInv.describeKey(), lBAAirlineCode);
  BOOST_CHECK_MESSAGE (lBAInv.describeKey() == lBAAirlineCode,
                       "The inventory key, '" << lBAInv.describeKey()
                       << "', should be equal to '" << lBAAirlineCode
                       << "', but is not");

  // Create an Inventory for AF
  const stdair::AirlineCode_T lAFAirlineCode ("AF");
  const stdair::InventoryKey lAFKey (lAFAirlineCode);
  myprovider::Inventory& lAFInv =
    stdair::FacBom<myprovider::Inventory>::instance().create (lAFKey);
  stdair::FacBomManager::addToList (lBomRoot, lAFInv);

  BOOST_CHECK_EQUAL (lAFInv.describeKey(), lAFAirlineCode);
  BOOST_CHECK_MESSAGE (lAFInv.describeKey() == lAFAirlineCode,
                       "The inventory key, '" << lAFInv.describeKey()
                       << "', should be equal to '" << lAFAirlineCode
                       << "', but is not");

  // Browse the inventories
  const myprovider::InventoryList_T& lInventoryList =
      stdair::BomManager::getList<myprovider::Inventory> (lBomRoot);
  const std::string lInventoryKeyArray[2] = {lBAAirlineCode, lAFAirlineCode};
```

```
  short idx = 0;
  for (myprovider::InventoryList_T::const_iterator itInv =
         lInventoryList.begin(); itInv != lInventoryList.end();
       ++itInv, ++idx) {
    const myprovider::Inventory* lInv_ptr = *itInv;
    BOOST_REQUIRE (lInv_ptr != NULL);

    BOOST_CHECK_EQUAL (lInventoryKeyArray[idx], lInv_ptr->describeKey());
    BOOST_CHECK_MESSAGE (lInventoryKeyArray[idx] == lInv_ptr->describeKey(),
                         "They inventory key, '" << lInventoryKeyArray[idx]
                         << "', does not match that of the Inventory object: '"
                         << lInv_ptr->describeKey() << "'");
  }
}

BOOST_AUTO_TEST_CASE (bom_structure_serialisation_test) {

  // Backup (thanks to Boost.Serialisation) file
  const std::string lBackupFilename = "StandardAirlineITTestSuite_serial.txt";

  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_serial.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // Retrieve (a reference on) the top of the persistent BOM tree
  stdair::BomRoot& lPersistentBomRoot = stdairService.getPersistentBomRoot();

  // DEBUG: Display the whole BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG (lCSVDump);

  // Clone the persistent BOM
  stdairService.clonePersistentBom ();

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string lBAInvKeyStr ("BA");
  stdair::Inventory* lBAInv_ptr =
    lPersistentBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key.");

  BOOST_REQUIRE_MESSAGE (lBAInv_ptr != NULL,
                         "An Inventory object should exist with the key, '"
                         << lBAInvKeyStr << "'.");

  // create and open a character archive for output
  std::ofstream ofs (lBackupFilename.c_str());

  // save data to archive
  {
    boost::archive::text_oarchive oa (ofs);
    // write class instance to archive
```

```
    oa << lPersistentBomRoot;
    // archive and stream closed when destructors are called
  }

  // ... some time later restore the class instance to its orginal state
  stdair::BomRoot& lRestoredBomRoot =
    stdair::FacBom<stdair::BomRoot>::instance().create();
  {
    // create and open an archive for input
    std::ifstream ifs (lBackupFilename.c_str());
    boost::archive::text_iarchive ia(ifs);
    // read class state from archive
    ia >> lRestoredBomRoot;
    // archive and stream closed when destructors are called
  }

  // DEBUG: Display the whole restored BOM tree
  const std::string& lRestoredCSVDump =
    stdairService.csvDisplay(lRestoredBomRoot);
  STDAIR_LOG_DEBUG (lRestoredCSVDump);

  // Retrieve the BomRoot key, and compare it to the expected one
  const std::string& lBomRootKeyStr = lRestoredBomRoot.describeKey();
  const std::string lBomRootString (" -- ROOT -- ");

  // DEBUG
  STDAIR_LOG_DEBUG ("The BOM root key is '" << lBomRootKeyStr
                    << "'. It should be equal to '" << lBomRootString << "'");

  BOOST_CHECK_EQUAL (lBomRootKeyStr, lBomRootString);
  BOOST_CHECK_MESSAGE (lBomRootKeyStr == lBomRootString,
                       "The BOM root key, '" << lBomRootKeyStr
                       << "', should be equal to '" << lBomRootString
                       << "', but is not.");

  // Retrieve the Inventory
  stdair::Inventory* lRestoredBAInv_ptr =
    lRestoredBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the restored BOM root.");

  BOOST_CHECK_MESSAGE (lRestoredBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the restored BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

BOOST_AUTO_TEST_CASE (bom_structure_clone_test) {

  // Output log File
  const std::string lLogFilename ("StandardAirlineITTestSuite_clone.log");

  // Set the log parameters
  std::ofstream logOutputFile;

  // Open and clean the log outputfile
  logOutputFile.open (lLogFilename.c_str());
  logOutputFile.clear();

  // Initialise the stdair BOM
  const stdair::BasLogParams lLogParams (stdair::LOG::DEBUG, logOutputFile);
  stdair::STDAIR_Service stdairService (lLogParams);

```

```
  // Build a sample BOM tree
  stdairService.buildSampleBom();

  // Retrieve (a constant reference on) the top of the persistent BOM tree
  const stdair::BomRoot& lPersistentBomRoot =
    stdairService.getPersistentBomRoot();

  // DEBUG: Display the whole persistent BOM tree
  const std::string& lCSVDump = stdairService.csvDisplay ();
  STDAIR_LOG_DEBUG ("Display the persistent BOM tree.");
  STDAIR_LOG_DEBUG (lCSVDump);

  // Clone the persistent BOM
  stdairService.clonePersistentBom ();

  // Retrieve (a reference on) the top of the clone BOM tree
  stdair::BomRoot& lCloneBomRoot = stdairService.getBomRoot();

  // DEBUG: Display the clone BOM tree after the clone process.
  const std::string& lAfterCloneCSVDump =
    stdairService.csvDisplay(lCloneBomRoot);
  STDAIR_LOG_DEBUG ("Display the clone BOM tree after the clone process.");
  STDAIR_LOG_DEBUG (lAfterCloneCSVDump);

  // Retrieve the clone BomRoot key, and compare it to the persistent BomRoot
  // key.
  const std::string& lCloneBomRootKeyStr = lCloneBomRoot.describeKey();
  const std::string& lPersistentBomRootKeyStr =
    lPersistentBomRoot.describeKey();

  // DEBUG
  STDAIR_LOG_DEBUG ("The clone BOM root key is '" << lCloneBomRootKeyStr
                    << "'. It should be equal to '"
                    << lPersistentBomRootKeyStr << "'");

  BOOST_CHECK_EQUAL (lCloneBomRootKeyStr, lPersistentBomRootKeyStr);
  BOOST_CHECK_MESSAGE (lCloneBomRootKeyStr == lPersistentBomRootKeyStr,
                       "The clone BOM root key, '" << lCloneBomRootKeyStr
                       << "', should be equal to '" << lPersistentBomRootKeyStr
                       << "', but is not.");

  // Retrieve the BA inventory in the clone BOM root
  const std::string lBAInvKeyStr ("BA");
  stdair::Inventory* lCloneBAInv_ptr =
    lCloneBomRoot.getInventory (lBAInvKeyStr);

  // DEBUG
  STDAIR_LOG_DEBUG ("There should be an Inventory object corresponding to the '"
                    << lBAInvKeyStr << "' key in the clone BOM root.");

  BOOST_CHECK_MESSAGE (lCloneBAInv_ptr != NULL,
                       "An Inventory object should exist with the key, '"
                       << lBAInvKeyStr << "' in the clone BOM root.");

  // Close the Log outputFile
  logOutputFile.close();
}

// End the test suite
BOOST_AUTO_TEST_SUITE_END()

#else // BOOST_VERSION >= 103900
boost_utf::test_suite* init_unit_test_suite (int, char* []) {
  boost_utf::test_suite* test = BOOST_TEST_SUITE ("Unit test example 1");
  return test;
}
#endif // BOOST_VERSION >= 103900
```

00427

33.649 test/stdair/StdairTestLib.hpp File Reference

```
#include <string>
#include <sstream>
```

Classes

- struct stdair_test::BookingClass
- struct stdair_test::Cabin

Namespaces

- namespace stdair_test

33.650 test/stdair/StdairTestLib.hpp

```
#ifndef __STDAIR_TST_STDAIR_TEST_LIB_HPP
#define __STDAIR_TST_STDAIR_TEST_LIB_HPP

// ////////////////////////////////////////////////////////////////////
// Import section
// ////////////////////////////////////////////////////////////////////
#include <string>
#include <sstream>

namespace stdair_test {

  struct BookingClass {
    std::string _classCode;
    BookingClass (const std::string& iClassCode)
      : _classCode (iClassCode) {
    }

    std::string toString() const {
      std::ostringstream oStr;
      oStr << _classCode;
      return oStr.str();
    }
  };

  struct Cabin {
    BookingClass _bookingClass;
    Cabin (const BookingClass& iBkgClass)
      : _bookingClass (iBkgClass) {
    }

    std::string toString() const {
      std::ostringstream oStr;
      oStr << _bookingClass._classCode;
      return oStr.str();
    }

    typedef BookingClass child;
  };
}

#endif // __STDAIR_TST_STDAIR_TEST_LIB_HPP
```