

# globus callout

## 2.2

Generated by Doxygen 1.8.0

Tue May 1 2012 17:29:04

## Contents

<b>1</b>	<b>Globus Callout API</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>2</b>
2.1	Modules . . . . .	2
<b>3</b>	<b>Module Documentation</b>	<b>2</b>
3.1	Activation . . . . .	2
3.1.1	Detailed Description . . . . .	2
3.1.2	Define Documentation . . . . .	2
3.2	Callout Handle Operations . . . . .	3
3.2.1	Detailed Description . . . . .	3
3.2.2	Typedef Documentation . . . . .	3
3.2.3	Function Documentation . . . . .	3
3.3	Callout Configuration . . . . .	4
3.3.1	Detailed Description . . . . .	4
3.3.2	Function Documentation . . . . .	4
3.4	Callout Invocation . . . . .	6
3.4.1	Detailed Description . . . . .	6
3.4.2	Typedef Documentation . . . . .	6
3.4.3	Function Documentation . . . . .	6
3.5	Globus Callout Constants . . . . .	7
3.5.1	Enumeration Type Documentation . . . . .	7

## 1 Globus Callout API

This API is intended to ease integration of configurable callouts into the Globus Toolkit and to provide a platform independent way of dealing with runtime loadable functions. It (hopefully) achieves this goal by providing the following functionality:

- It provides a function for reading callout configuration files. Files are assumed to have the following format:
  - Anything after a '#' is assumed to be a comment
  - Blanks lines are ignored
  - Lines specifying callouts have the format abstract type library symbol where "abstract type" denotes the type of callout, e.g. `globus_gram_jobmanager_authz`, "library" denotes the library the callout can be found in and "symbol" denotes the function name of the callout.

It provides a API function for registering callouts

- All callouts are assumed to have the function signature `globus_result_t callout_func(va_list ap)`
- It provides a function for calling a callout given a abstract type. If multiple callouts are defined for the same abstract type then all callouts for the abstract type will be called. Implementers should not rely on any

correlation between the order of configuration and the order of invocation of callouts of the same abstract type.

Any program that uses Globus Callout functions must include "globus\_callout.h".

## 2 Module Index

### 2.1 Modules

Here is a list of all modules:

<b>Activation</b>	<b>2</b>
<b>Callout Handle Operations</b>	<b>3</b>
<b>Callout Configuration</b>	<b>4</b>
<b>Callout Invocation</b>	<b>6</b>
<b>Globus Callout Constants</b>	<b>7</b>

## 3 Module Documentation

### 3.1 Activation

#### Defines

- `#define GLOBUS_CALLOUT_MODULE`

#### 3.1.1 Detailed Description

Globus Callout API uses standard Globus module activation and deactivation. Before any Globus Callout API functions are called, the following function must be called:

```
globus_module_activate(GLOBUS_CALLOUT_MODULE)
```

This function returns `GLOBUS_SUCCESS` if Globus Callout API was successfully initialized, and you are therefore allowed to subsequently call Globus Callout API functions. Otherwise, an error code is returned, and Globus GSI Credential functions should not be subsequently called. This function may be called multiple times.

To deactivate Globus Callout API, the following function must be called:

```
globus_module_deactivate(GLOBUS_CALLOUT_MODULE)
```

This function should be called once for each time Globus Callout API was activated.

#### 3.1.2 Define Documentation

##### 3.1.2.1 `#define GLOBUS_CALLOUT_MODULE`

Module descriptor.

## 3.2 Callout Handle Operations

### Typedefs

- typedef struct  
globus\_i\_callout\_handle\_s \* **globus\_callout\_handle\_t**

### Initialize Handle

- globus\_result\_t **globus\_callout\_handle\_init** (globus\_callout\_handle\_t \*handle)

### Destroy Handle

- globus\_result\_t **globus\_callout\_handle\_destroy** (globus\_callout\_handle\_t handle)

#### 3.2.1 Detailed Description

Initialize and Destory a Globus Callout Handle structure. This section defines operations for initializing and destroying Globus Callout Handle structure.

#### 3.2.2 Typedef Documentation

##### 3.2.2.1 typedef struct globus\_i\_callout\_handle\_s\* globus\_callout\_handle\_t

Callout handle type definition.

#### 3.2.3 Function Documentation

##### 3.2.3.1 globus\_result\_t globus\_callout\_handle\_init ( globus\_callout\_handle\_t \* *handle* )

Initialize a Globus Callout Handle.

#### Parameters

<i>handle</i>	Pointer to the handle that is to be initialized
---------------	---

#### Returns

GLOBUS\_SUCCESS if successful A Globus error object on failure: GLOBUS\_CALLOUT\_ERROR\_WITH\_H-ASHTABLE

##### 3.2.3.2 globus\_result\_t globus\_callout\_handle\_destroy ( globus\_callout\_handle\_t *handle* )

Destroy a Globus Callout Handle.

#### Parameters

<i>handle</i>	The handle that is to be destroyed
---------------	------------------------------------

#### Returns

GLOBUS\_SUCCESS

## 3.3 Callout Configuration

### Configure Callouts

- `globus_result_t globus_callout_read_config (globus_callout_handle_t handle, char *filename)`
- `globus_result_t globus_callout_register (globus_callout_handle_t handle, char *type, char *library, char *symbol)`

#### 3.3.1 Detailed Description

Functions for registering callouts. This section defines operations for registering callouts. Callouts may be registered either through a configuration file or through calls to `globus_callout_register`.

#### 3.3.2 Function Documentation

##### 3.3.2.1 `globus_result_t globus_callout_read_config ( globus_callout_handle_t handle, char * filename )`

Read callout configuration from file.

This function read a configuration file with the following format:

- Anything after a '#' is assumed to be a comment
- Blank lines are ignored
- Lines specifying callouts have the format `abstract type library symbol` where "abstract type" denotes the type of callout, e.g. `globus_gram_jobmanager_authz`, "library" denotes the library the callout can be found in and "symbol" denotes the function name of the callout. The library argument can be specified in two forms, `libfoo` or `libfoo_<flavor>`. When using the former version the current flavor will automatically be added to the library name.

#### Parameters

<i>handle</i>	The handle that is to be configured
<i>filename</i>	The file to read configuration from

#### Returns

`GLOBUS_SUCCESS` A Globus error object on failure: `GLOBUS_CALLOUT_ERROR_OPENING_CONF_FILE` `GLOBUS_CALLOUT_ERROR_PARSING_CONF_FILE` `GLOBUS_CALLOUT_ERROR_WITH_HASHTABLE` `GLOBUS_CALLOUT_ERROR_OUT_OF_MEMORY`

##### 3.3.2.2 `globus_result_t globus_callout_register ( globus_callout_handle_t handle, char * type, char * library, char * symbol )`

Register callout configuration

This function registers a callout type in the given handle.

#### Parameters

<i>handle</i>	The handle that is to be configured
<i>type</i>	The abstract type of the callout
<i>library</i>	The location of the library containing the callout
<i>symbol</i>	The symbol (ie function name) for the callout

## Returns

GLOBUS\_SUCCESS A Globus error object on failure: GLOBUS\_CALLOUT\_ERROR\_WITH\_HASHTABLE  
GLOBUS\_CALLOUT\_ERROR\_OUT\_OF\_MEMORY

## 3.4 Callout Invocation

### Typedefs

- `typedef globus_result_t(* globus_callout_function_t )(va_list ap)`

### Invoking Callouts

- `globus_result_t globus_callout_call_type (globus_callout_handle_t handle, char *type,...)`

#### 3.4.1 Detailed Description

Functions for invoking callouts. This section defines a operation for invoking callouts by their abstract type.

#### 3.4.2 Typedef Documentation

##### 3.4.2.1 `typedef globus_result_t(* globus_callout_function_t)(va_list ap)`

Callout function type definition.

#### 3.4.3 Function Documentation

##### 3.4.3.1 `globus_result_t globus_callout_call_type ( globus_callout_handle_t handle, char * type, ... )`

Call a callout of specified abstract type

This function looks up the callouts corresponding to the given type and invokes them with the passed arguments.

If a invoked callout returns an error it will be chained to a error of the type `GLOBUS_CALLOUT_ERROR_CALLOUT_ERROR` and no more callouts will be called.

#### Parameters

<i>handle</i>	A configured callout handle
<i>type</i>	The abstract type of the callout that is to be invoked

#### Returns

`GLOBUS_SUCCESS` A Globus error object on failure: `GLOBUS_CALLOUT_ERROR_TYPE_NOT_REGISTERED` `GLOBUS_CALLOUT_ERROR_CALLOUT_ERROR` `GLOBUS_CALLOUT_ERROR_WITH_DL` `GLOBUS_CALLOUT_ERROR_WITH_HASHTABLE` `GLOBUS_CALLOUT_ERROR_OUT_OF_MEMORY`

## 3.5 Globus Callout Constants

### Enumerations

- enum **globus\_callout\_error\_t** {  
    **GLOBUS\_CALLOUT\_ERROR\_SUCCESS** = 0,  
    **GLOBUS\_CALLOUT\_ERROR\_WITH\_HASHTABLE** = 1,  
    **GLOBUS\_CALLOUT\_ERROR\_OPENING\_CONF\_FILE** = 2,  
    **GLOBUS\_CALLOUT\_ERROR\_PARSING\_CONF\_FILE** = 3,  
    **GLOBUS\_CALLOUT\_ERROR\_WITH\_DL** = 4,  
    **GLOBUS\_CALLOUT\_ERROR\_OUT\_OF\_MEMORY** = 5,  
    **GLOBUS\_CALLOUT\_ERROR\_TYPE\_NOT\_REGISTERED** = 6,  
    **GLOBUS\_CALLOUT\_ERROR\_CALLOUT\_ERROR** = 7,  
    **GLOBUS\_CALLOUT\_ERROR\_LAST** = 8 }

### 3.5.1 Enumeration Type Documentation

#### 3.5.1.1 enum **globus\_callout\_error\_t**

Globus Callout Error codes.

Enumerator:

- GLOBUS\_CALLOUT\_ERROR\_SUCCESS** Success - never used.
- GLOBUS\_CALLOUT\_ERROR\_WITH\_HASHTABLE** Hash table operation failed.
- GLOBUS\_CALLOUT\_ERROR\_OPENING\_CONF\_FILE** Failed to open configuration file.
- GLOBUS\_CALLOUT\_ERROR\_PARSING\_CONF\_FILE** Failed to parse configuration file.
- GLOBUS\_CALLOUT\_ERROR\_WITH\_DL** Dynamic library operation failed.
- GLOBUS\_CALLOUT\_ERROR\_OUT\_OF\_MEMORY** Out of memory.
- GLOBUS\_CALLOUT\_ERROR\_TYPE\_NOT\_REGISTERED** The abstract type could not be found.
- GLOBUS\_CALLOUT\_ERROR\_CALLOUT\_ERROR** The callout itself returned a error.
- GLOBUS\_CALLOUT\_ERROR\_LAST** Last marker - never used.



## Index

- Activation, 1
  - GLOBUS\_CALLOUT\_MODULE, 2
- Callout Configuration, 4
  - globus\_callout\_read\_config, 4
  - globus\_callout\_register, 4
- Callout Handle Operations, 3
  - globus\_callout\_handle\_destroy, 3
  - globus\_callout\_handle\_init, 3
  - globus\_callout\_handle\_t, 3
- Callout Invocation, 6
  - globus\_callout\_call\_type, 6
  - globus\_callout\_function\_t, 6
- GLOBUS\_CALLOUT\_ERROR\_CALLOUT\_ERROR
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_LAST
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_OPENING\_CONF\_FILE
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_OUT\_OF\_MEMORY
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_PARSING\_CONF\_FILE
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_SUCCESS
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_TYPE\_NOT\_REGISTERED
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_WITH\_DL
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_ERROR\_WITH\_HASHTABLE
  - Globus Callout Constants, 7
- GLOBUS\_CALLOUT\_MODULE
  - Activation, 2
- Globus Callout Constants, 7
  - GLOBUS\_CALLOUT\_ERROR\_CALLOUT\_ERROR, 7
  - GLOBUS\_CALLOUT\_ERROR\_LAST, 7
  - GLOBUS\_CALLOUT\_ERROR\_OPENING\_CONF\_FILE, 7
  - GLOBUS\_CALLOUT\_ERROR\_OUT\_OF\_MEMORY, 7
  - GLOBUS\_CALLOUT\_ERROR\_PARSING\_CONF\_FILE, 7
  - GLOBUS\_CALLOUT\_ERROR\_SUCCESS, 7
  - GLOBUS\_CALLOUT\_ERROR\_TYPE\_NOT\_REGISTERED, 7
  - GLOBUS\_CALLOUT\_ERROR\_WITH\_DL, 7
  - GLOBUS\_CALLOUT\_ERROR\_WITH\_HASHTABLE, 7
  - globus\_callout\_error\_t, 7
  - globus\_callout\_call\_type
    - Callout Invocation, 6
  - globus\_callout\_error\_t
    - Globus Callout Constants, 7
  - globus\_callout\_function\_t
    - Callout Invocation, 6
  - globus\_callout\_handle\_destroy
    - Callout Handle Operations, 3
  - globus\_callout\_handle\_init
    - Callout Handle Operations, 3
  - globus\_callout\_handle\_t
    - Callout Handle Operations, 3
  - globus\_callout\_read\_config
    - Callout Configuration, 4
  - globus\_callout\_register
    - Callout Configuration, 4