

# globus callout Reference Manual

## 0.7

Generated by Doxygen 1.2.18

Fri Jun 26 15:34:59 2009

## Contents

<a href="#">1 Globus Callout API</a>	1
<a href="#">2 globus callout Module Index</a>	1
<a href="#">3 globus callout Module Documentation</a>	2

## 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
- Blank lines are ignored
- Lines specifying callouts have the format `abstract type library symbol` where "abstract type" denotes the type of callout, e.g. `globusgram.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_t callout_func(valist 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 "globuscallout.h".

## 2 globus callout Module Index

### 2.1 globus callout Modules

Here is a list of all modules:

<a href="#">Activation</a>	2
<a href="#">Callout Handle Operations</a>	2
<a href="#">Callout Configuration</a>	3
<a href="#">Callout Invocation</a>	4
<a href="#">Globus Callout Constants</a>	5

## 3 globus callout Module Documentation

### 3.1 Activation

Globus Callout API uses standard Globus module activation and deactivation.

De nes

```
#de ne 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 De ne Documentation

##### 3.1.2.1 #de ne GLOBUSCALLOUT\_MODULE

Module descriptor.

## 3.2 Callout Handle Operations

Initialize and Destory a Globus Callout Handle structure.

Initialize Handle

```
globus_result_t globus_callout_handleinit (globus_callout_handle_t handle)
```

Destroy Handle

```
globus_result_t globus_callout_handledestroy(globus_callout_handle_t handle)
```

Typedefs

```
typedef globus_callout_handles globus_callout_handle_t
```

### 3.2.1 Detailed Description

Initialize and Destroy a Globus Callout Handle structure.

This section describes operations for initializing and destroying Globus Callout Handle structure.

### 3.2.2 Typedef Documentation

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

Callout handle type definition.

### 3.2.3 Function Documentation

#### 3.2.3.1 globusresult\_t globus\_callout\_handle\_init (globus\_callout\_handle\_t handle)

Initialize a Globus Callout Handle.

Parameters:

handle Pointer to the handle that is to be initialized

Returns:

GLOBUS\_SUCCESS if successful A Globus error object on failure: GLOBUS\_CALLOUT\_ERRORWITH\_HASHTABLE

#### 3.2.3.2 globusresult\_t globus\_callout\_handle\_destroy (globus\_callout\_handle\_t handle)

Destroy a Globus Callout Handle.

Parameters:

handle The handle that is to be destroyed

Returns:

GLOBUS\_SUCCESS

## 3.3 Callout Configuration

Functions for registering callouts.

Configure Callouts

```
globusresult_t globus_callout_readconfig (globus_callout_handle_t handle, char lename)
globusresult_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 describes operations for registering callouts. Callouts may be registered either through a configuration file or through calls to globusallout.register.

## 3.3.2 Function Documentation

## 3.3.2.1 globusresult\_t globus\_callout\_read\_config (globus\_callout\_handle\_t handle, char \* filename)

Read callout configuration from file.

This function reads 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. globusgramjobmanagerauthz, "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.< avor> . When using the former version the current avor will automatically be added to the library name.

Parameters:

handle The handle that is to be configured

filename The file to read configuration from

Returns:

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

## 3.3.2.2 globusresult\_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:

handle The handle that is to be configured

type The abstract type of the callout

library The location of the library containing the callout

symbol 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

Functions for invoking callouts.

Invoking Callouts

globusresult\_t globus\_callout\_call\_type (globus\_callout\_handle\_t handle, char \* type, ...)

## Typedefs

```
typedef globusresult_t( globuscalloutfunction_t )(va_list ap)
```

### 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 globusresult\_t( globus\_callout\_function\_t)( va\_list ap)

Callout function type definition.

### 3.4.3 Function Documentation

### 3.4.3.1 globusresult\_t globus\_callout\_call\_type (globus\_callout\_handle\_t handle, char type ...)

Call a callout of speci ed 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_ERRORCALLOUT_ERROR` and no more callouts will be called.

Parameters:

handle A configured callout handle

type The abstract type of the callout that is to be invoked

Returns:

GLOBAL SUCCESS A Globus error object on failure: GLOBALSCALLOUT\_ERRORTYPE\_NOT-  
REGISTERED GLOBALSCALLOUT\_ERRORGLOBALSCALLOUT\_ERRORGLOBALSCALLOUT\_ERRORWITH-  
DLGLOBALSCALLOUT\_ERRORWITH\_HASHTABLEGLOBALSCALLOUT\_ERROROUT OF MEMORY

### 3.5 Globus Callout Constants

## Enumerations

```
enum globuscallout_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 globuscallout\_error\_t

## Globus Callout Error codes.

Enumeration values:

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 an error.  
GLOBUS\_CALLOUT\_ERROR\_LAST Last marker - never used.

---

## Index

Activation, [2](#)

Callout Con guration, [3](#)

Callout Handle Operation, [3](#)

Callout Invocation, [4](#)

Globus Callout Constant, [5](#)

globus.callout.activation

    GLOBUS\_CALLOUT\_MODULE, [2](#)

globus.callout.call

    globus.callout.call.type, [5](#)

    globus.callout.function.t, [5](#)

globus.callout.call.type

    globus.callout.call, [5](#)

globus.callout.con g

    globus.callout.read.con g, [4](#)

    globus.callout.register, [4](#)

globus.callout.constants

    GLOBUS\_CALLOUT\_ERRORCALLOUT\_-  
    ERROR, [6](#)

    GLOBUS\_CALLOUT\_ERRORLAST, [6](#)

    GLOBUS\_CALLOUT\_ERROROPENING\_-  
    CONF\_FILE, [6](#)

    GLOBUS\_CALLOUT\_ERROROUT\_OF\_-  
    MEMORY, [6](#)

    GLOBUS\_CALLOUT\_ERRORPARSING\_-  
    CONF\_FILE, [6](#)

    GLOBUS\_CALLOUT\_ERRORSUCCESS, [5](#)

    GLOBUS\_CALLOUT\_ERRORTYPE\_NOT\_-  
    REGISTERED, [6](#)

    GLOBUS\_CALLOUT\_ERRORWITH\_DL, [6](#)

    GLOBUS\_CALLOUT\_ERRORWITH\_-  
    HASHTABLE, [5](#)

globus.callout.constants

    globus.callout.error.t, [5](#)

GLOBUS\_CALLOUT\_ERRORCALLOUT\_ERROR

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERRORLAST

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERROROPENING.CONF\_-  
FILE

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERROROUT\_OF\_MEMORY

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERRORPARSING.CONF\_-  
FILE

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERRORSUCCESS

    globus.callout.constants, [5](#)

globus.callout.error.t

    globus.callout.constants, [5](#)

GLOBUS\_CALLOUT\_ERRORTYPE\_NOT\_-  
REGISTERED

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERRORWITH\_DL

    globus.callout.constants, [6](#)

GLOBUS\_CALLOUT\_ERRORWITH\_-  
HASHTABLE

    globus.callout.constants, [5](#)

globus.callout.function.t

    globus.callout.call, [5](#)

globus.callout.handle

    globus.callout.handledestroy, [3](#)

    globus.callout.handleinit, [3](#)

    globus.callout.handlet, [3](#)

globus.callout.handledestroy

    globus.callout.handle, [3](#)

globus.callout.handleinit

    globus.callout.handle, [3](#)

globus.callout.handlet

    globus.callout.handle, [3](#)

GLOBUS\_CALLOUT\_MODULE

    globus.callout.activation, [2](#)

globus.callout.read.con g

    globus.callout.con g, [4](#)

globus.callout.register

    globus.callout.con g, [4](#)