globus callout Reference Manual

0.7

Generated by Doxygen 1.2.18

Tue Aug 11 22:33:48 2009

Contents

1 Globus Callout API

This API is intended to ease integration of con gurable 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 con guration les. 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 de ned 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 con guration 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 globus callout Module Index

2.1 globus callout Modules

Here is a list of all modules:

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 GLOBUS_CALLOUT_MODULE

Module descriptor.

3.2 Callout Handle Operations

Initialize and Destory a Globus Callout Handle structure.

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)

Typedefs

- typedef globus_i_callout_handle_s * globus_callout_handle_t

3.2.1 Detailed Description

Initialize and Destory a Globus Callout Handle structure.

This section de nes 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 de nition.

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:
handle Pointer to the handle that is to be initialized

Returns:
GLOBUS_SUCCESS if successful A Globus error object on failure: GLOBUS_CALLOUT_ERROR_WITH_-HASHTABLE

3.2.3.2 globus_result_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 Con guration

Functions for registering callouts.

Con gure Callouts

- globus_result_t globus_callout_read_con g (globus_callout_handle_t handle, char * lename)
- 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 de nes operations for registering callouts. Callouts may be registered either through a con guration le or through calls to globus_callout_register.

3.3.2 Function Documentation

3.3.2.1 globus_result_t globus_callout_read_con g (globus_callout_handle_t handle, char lename)

Read callout con guration from le.

This function read a con guration le with 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. The library argument can be speci ed 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 con gured
lename	The le to read con guration 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 con guration.

This function registers a callout type in the given handle.

Parameters:

handle	The handle that is to be con gured
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

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

Typedefs

- typedef globus_result_t(globus_callout_function_t)(va_list ap)

3.4.1 Detailed Description

Functions for invoking callouts.

This section de nes 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 de nition.

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 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_ERROR_CALLOUT_ERROR and no more callouts will be called.

Parameters:

handle A con gured callout handle

type 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.

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 con guration le.

GLOBUS_CALLOUT_ERROR_PARSING_CONF_FILE Failed to parse con guration le.

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