

# globus gsi sysconfig

## 5.2

Generated by Doxygen 1.7.5

Mon May 14 2012 10:31:29

# Contents

<b>1</b>	<b>Globus GSI System Config API</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>1</b>
2.1	Modules . . . . .	1
<b>3</b>	<b>Module Documentation</b>	<b>2</b>
3.1	Defines . . . . .	2
3.1.1	Detailed Description . . . . .	2
3.1.2	Define Documentation . . . . .	2
3.2	Functions for UNIX platforms . . . . .	6
3.2.1	Detailed Description . . . . .	8
3.2.2	Function Documentation . . . . .	8
3.3	Functions for Win32 platforms . . . . .	16
3.3.1	Detailed Description . . . . .	17
3.3.2	Function Documentation . . . . .	17
3.4	Functions for all platforms . . . . .	24
3.4.1	Detailed Description . . . . .	24
3.4.2	Function Documentation . . . . .	24
3.5	Activation . . . . .	25
3.5.1	Detailed Description . . . . .	25
3.5.2	Define Documentation . . . . .	25
3.6	Datatypes . . . . .	26
3.6.1	Enumeration Type Documentation . . . . .	26

## 1 Globus GSI System Config API

This API provides helper functions for detecting installation and environment specific settings applicabale to GSI. It also servers as a abstraction layer for OS specific programming details. This is achieves by defining preprocessor symbols that point at the correct platform specific function. **You should never use the platform specific functions directly..** Any program that uses Globus GSI System Config functions must include "globus\_gsi\_system\_config.h".

## 2 Module Index

### 2.1 Modules

Here is a list of all modules:

<b>Defines</b>	<b>2</b>
<b>Functions for UNIX platforms</b>	<b>6</b>

<b>Functions for Win32 platforms</b>	<b>16</b>
<b>Functions for all platforms</b>	<b>24</b>
<b>Activation</b>	<b>25</b>
<b>Datatypes</b>	<b>26</b>

## 3 Module Documentation

### 3.1 Defines

#### Defines

- `#define GLOBUS_GSI_SYSCONFIG_SET_KEY_PERMISSIONS`
- `#define GLOBUS_GSI_SYSCONFIG_GET_HOME_DIR`
- `#define GLOBUS_GSI_SYSCONFIG_CHECK_KEYFILE`
- `#define GLOBUS_GSI_SYSCONFIG_CHECK_CERTFILE`
- `#define GLOBUS_GSI_SYSCONFIG_FILE_EXISTS`
- `#define GLOBUS_GSI_SYSCONFIG_DIR_EXISTS`
- `#define GLOBUS_GSI_SYSCONFIG_GET_CERT_DIR`
- `#define GLOBUS_GSI_SYSCONFIG_GET_USER_CERT_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_HOST_CERT_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_SERVICE_CERT_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_PROXY_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_SIGNING_POLICY_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_CA_CERT_FILES`
- `#define GLOBUS_GSI_SYSCONFIG_GET_CURRENT_WORKING_DIR`
- `#define GLOBUS_GSI_SYSCONFIG_MAKE_ABSOLUTE_PATH_FOR_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_SPLIT_DIR_AND_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_REMOVE_ALL_OWNED_FILES`
- `#define GLOBUS_GSI_SYSCONFIG_GET_GRIDMAP_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_AUTHZ_CONF_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_GAA_CONF_FILENAME`
- `#define GLOBUS_GSI_SYSCONFIG_IS_SUPERUSER`
- `#define GLOBUS_GSI_SYSCONFIG_GET_USER_ID_STRING`
- `#define GLOBUS_GSI_SYSCONFIG_GET_PROC_ID_STRING`
- `#define GLOBUS_GSI_SYSCONFIG_GET_USERNAME`
- `#define GLOBUS_GSI_SYSCONFIG_GET_UNIQUE_PROXY_FILENAME`

#### 3.1.1 Detailed Description

These precompiler defines allow for a platform (ie Win32 vs UNIX) independent API.

#### 3.1.2 Define Documentation

##### 3.1.2.1 `#define GLOBUS_GSI_SYSCONFIG_SET_KEY_PERMISSIONS`

Set the correct file permissions on a private key.

See `globus_gsi_sysconfig_set_key_permissions_unix()` (p. 8) and `globus_gsi_sysconfig_set_key_permissions_win32()` (p. 17)

### 3.1.2.2 #define GLOBUS\_GSI\_SYSCONFIG.GET\_HOME\_DIR

Get the current users home directory

See **globus\_gsi\_sysconfig\_get\_home\_dir\_unix()** (p. 10) and **globus\_gsi\_sysconfig\_get\_home\_dir\_win32()**

### 3.1.2.3 #define GLOBUS\_GSI\_SYSCONFIG.CHECK.KEYFILE

Check for the correct file permissions on a private key.

See **globus\_gsi\_sysconfig\_check\_keyfile\_unix()** (p. 10) and **globus\_gsi\_sysconfig\_check\_keyfile\_win32()**

### 3.1.2.4 #define GLOBUS\_GSI\_SYSCONFIG.CHECK.CERTFILE

Check for the correct file permissions on a certificate.

See **globus\_gsi\_sysconfig\_check\_certfile\_unix()** (p. 11) and **globus\_gsi\_sysconfig\_check\_certfile\_win32()**

### 3.1.2.5 #define GLOBUS\_GSI\_SYSCONFIG.FILE.EXISTS

Check whether a given file exists

See **globus\_gsi\_sysconfig\_file\_exists\_unix()** (p. 10) and **globus\_gsi\_sysconfig\_file\_exists\_win32()** (p. 18)

### 3.1.2.6 #define GLOBUS\_GSI\_SYSCONFIG.DIR.EXISTS

Check whether a given directory exists

See **globus\_gsi\_sysconfig\_dir\_exists\_unix()** (p. 10) and **globus\_gsi\_sysconfig\_dir\_exists\_win32()** (p. 18)

### 3.1.2.7 #define GLOBUS\_GSI\_SYSCONFIG.GET.CERT.DIR

Determine the location of the trusted certificates directory

See **globus\_gsi\_sysconfig\_get\_cert\_dir\_unix()** (p. 11) and **globus\_gsi\_sysconfig\_get\_cert\_dir\_win32()** (p. 19)

### 3.1.2.8 #define GLOBUS\_GSI\_SYSCONFIG.GET.USER.CERT.FILENAME

Determine the location of the users certificate and private key

See **globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_unix()** (p. 11) and **globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_win32()** (p. 19)

### 3.1.2.9 #define GLOBUS\_GSI\_SYSCONFIG.GET.HOST.CERT.FILENAME

Determine the location of the host certificate and private key

See **globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_unix()** (p. 12) and **globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_win32()** (p. 20)

### 3.1.2.10 #define GLOBUS\_GSI\_SYSCONFIG.GET.SERVICE.CERT.FILENAME

Determine the location of a service certificate and private key

See **globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_unix()** (p. 12) and **globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_win32()** (p. 20)

### 3.1.2.11 #define GLOBUS\_GSI\_SYSCONFIG.GET.PROXY.FILENAME

Determine the location of a proxy certificate and private key

See **globus\_gsi\_sysconfig\_get\_proxy\_filename\_unix()** (p. 13) and **globus\_gsi\_sysconfig\_get\_proxy\_**

**filename\_win32()** (p. 21)

#### 3.1.2.12 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_SIGNING\_POLICY\_FILENAME

Determine the name of the signing policy file for a given CA

See **globus\_gsi\_sysconfig\_get\_signing\_policy\_filename\_unix()** (p. 13) and **globus\_gsi\_sysconfig\_get\_signing\_policy\_filename\_win32()** (p. 23)

#### 3.1.2.13 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_CA\_CERT\_FILES

Get a list of trusted CA certificate filenames in a trusted CA certificate directory.

See **globus\_gsi\_sysconfig\_get\_ca\_cert\_files\_unix()** (p. 13) and **globus\_gsi\_sysconfig\_get\_ca\_cert\_files\_win32()** (p. 21)

#### 3.1.2.14 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_CURRENT\_WORKING\_DIR

Get the current working directory

See **globus\_gsi\_sysconfig\_get\_current\_working\_dir\_unix()** (p. 9) and **globus\_gsi\_sysconfig\_get\_current\_working\_dir\_win32()** (p. 18)

#### 3.1.2.15 #define GLOBUS\_GSI\_SYSCONFIG\_MAKE\_ABSOLUTE\_PATH\_FOR\_FILENAME

Prepend the current working directory to the given filename

See **globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_unix()** (p. 9) and **globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_win32()** (p. 18)

#### 3.1.2.16 #define GLOBUS\_GSI\_SYSCONFIG\_SPLIT\_DIR\_AND\_FILENAME

Split directory component of path from filename.

See **globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_unix()** (p. 9) and **globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_win32()** (p. 19)

#### 3.1.2.17 #define GLOBUS\_GSI\_SYSCONFIG\_REMOVE\_ALL\_OWNED\_FILES

Remove all proxies owned by current uid

See **globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_unix()** (p. 14) and **globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_win32()** (p. 22)

#### 3.1.2.18 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_GRIDMAP\_FILENAME

Determine the location of the grid map file.

See **globus\_gsi\_sysconfig\_get\_gridmap\_filename\_unix()** (p. 14) and **globus\_gsi\_sysconfig\_get\_gridmap\_filename\_win32()** (p. 22)

#### 3.1.2.19 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_AUTHZ\_CONF\_FILENAME

Determine the location of the authorization callout config file.

See **globus\_gsi\_sysconfig\_get\_authz\_conf\_filename\_unix()** (p. 14)

#### 3.1.2.20 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_GAA\_CONF\_FILENAME

Determine the location of the GAA callout config file.

See **globus\_gsi\_sysconfig\_get\_gaa\_conf\_filename\_unix()** (p. 15)

#### 3.1.2.21 #define GLOBUS\_GSI\_SYSCONFIG\_IS\_SUPERUSER

Determine whether the current user is the super user

See **globus\_gsi\_sysconfig\_is\_superuser\_unix()** (p. 14) and **globus\_gsi\_sysconfig\_is\_superuser\_win32()** (p. 23)

#### 3.1.2.22 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_USER\_ID\_STRING

Get the current UID in string form

See **globus\_gsi\_sysconfig\_get\_user\_id\_string\_unix()** (p. 8) and **globus\_gsi\_sysconfig\_get\_user\_id\_string\_win32()**

#### 3.1.2.23 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_PROC\_ID\_STRING

Get the current PID in string form

See **globus\_gsi\_sysconfig\_get\_proc\_id\_string\_unix()** (p. 9) and **globus\_gsi\_sysconfig\_get\_proc\_id\_string\_win32()**

#### 3.1.2.24 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_USERNAME

Get the current user name

See **globus\_gsi\_sysconfig\_get\_username\_unix()** (p. 8) and **globus\_gsi\_sysconfig\_get\_username\_win32()**

#### 3.1.2.25 #define GLOBUS\_GSI\_SYSCONFIG\_GET\_UNIQUE\_PROXY\_FILENAME

Generate a unique proxy file name

See **globus\_gsi\_sysconfig\_get\_unique\_proxy\_filename()** (p. 24)

## 3.2 Functions for UNIX platforms

### UNIX - Set Key Permissions

- globus\_result\_t **globus\_gsi\_sysconfig\_set\_key\_permissions\_unix** (char \*filename)

### UNIX - Get User ID

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_user\_id\_string\_unix** (char \*\*user\_id\_string)

### UNIX - Get Username

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_username\_unix** (char \*\*username)

### UNIX - Get Process ID

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_proc\_id\_string\_unix** (char \*\*proc\_id\_string)

### UNIX - Make Absolute Path

- globus\_result\_t **globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_unix** (char \*filename, char \*\*absolute\_path)

### UNIX - Split Directory and Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_unix** (char \*full\_filename, char \*\*dir\_string, char \*\*filename\_string)

### UNIX - Get Current Working Directory

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_current\_working\_dir\_unix** (char \*\*working\_dir)

### UNIX - Get HOME Directory

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_home\_dir\_unix** (char \*\*home\_dir)

### UNIX - File Exists

- globus\_result\_t **globus\_gsi\_sysconfig\_file\_exists\_unix** (const char \*filename)

### UNIX - Directory Exists

- globus\_result\_t **globus\_gsi\_sysconfig\_dir\_exists\_unix** (const char \*filename)

### UNIX - Check File Status for Key

- globus\_result\_t **globus\_gsi\_sysconfig\_check\_keyfile\_unix** (const char \*filename)

#### UNIX - Check File Status for Cert

- globus\_result\_t **globus\_gsi\_sysconfig\_check\_certfile\_unix** (const char \*filename)

#### UNIX - Get Trusted CA Cert Dir

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_cert\_dir\_unix** (char \*\*cert\_dir)

#### UNIX - Get User Certificate and Key Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_unix** (char \*\*user\_cert, char \*\*user\_key)

#### UNIX - Get Host Certificate and Key Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_unix** (char \*\*host\_cert, char \*\*host\_key)

#### UNIX - Get Service Certificate and Key Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_unix** (char \*service\_name, char \*\*service\_cert, char \*\*service\_key)

#### UNIX - Get Proxy Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_proxy\_filename\_unix** (char \*\*user\_proxy, **globus\_gsi\_proxy\_file\_type\_t** proxy\_file\_type)

#### UNIX - Get Signing Policy Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_signing\_policy\_filename\_unix** (X509\_NAME \*ca\_name, char \*cert\_dir, char \*\*signing\_policy\_filename)

#### UNIX - Get CA Cert Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_ca\_cert\_files\_unix** (char \*ca\_cert\_dir, globus\_fifo\_t \*ca\_cert\_list)

#### UNIX - Remove all proxies owned by current uid

- globus\_result\_t **globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_unix** (char \*default\_filename)

#### UNIX - Check if the current user is root

- globus\_result\_t **globus\_gsi\_sysconfig\_is\_superuser\_unix** (int \*is\_superuser)

#### UNIX - Get the path and file name of the grid map file

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_gridmap\_filename\_unix** (char \*\*filename)



UNIX - Get the path and file name of the authorization callback configuration file

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_authz\_conf\_filename\_unix** (char \*\*filename)
- globus\_result\_t **globus\_gsi\_sysconfig\_get\_authz\_lib\_conf\_filename\_unix** (char \*\*filename)

UNIX - Get the path and file name of the gaa configuration file

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_gaa\_conf\_filename\_unix** (char \*\*filename)

### 3.2.1 Detailed Description

These functions implement the UNIX version of the Globus GSI System Configuration API. **They should never be called directly, please use the provided platform independent defines.**

### 3.2.2 Function Documentation

#### 3.2.2.1 globus\_result\_t globus\_gsi\_sysconfig\_set\_key\_permissions\_unix ( char \* filename )

Set the file permissions of a file to read-write only by the user which are the permissions that should be set for all private keys.

##### Parameters

<i>filename</i>	
-----------------	--

##### Returns

GLOBUS\_SUCCESS or an error object id

#### 3.2.2.2 globus\_result\_t globus\_gsi\_sysconfig\_get\_user\_id\_string\_unix ( char \*\* user\_id\_string )

Get a unique string representing the current user.

This is just the uid converted to a string.

##### Parameters

<i>user_id_string</i>	A unique string representing the user
-----------------------	---------------------------------------

##### Returns

GLOBUS\_SUCCESS unless an error occurred

#### 3.2.2.3 globus\_result\_t globus\_gsi\_sysconfig\_get\_username\_unix ( char \*\* username )

Get the username of the current user.

##### Parameters

<i>username</i>	This parameter will contain the current user name upon a successful return. It is the users responsibility to free memory allocated for this return value.
-----------------	--

#### Returns

GLOBUS\_SUCCESS unless an error occurred

#### 3.2.2.4 globus\_result\_t globus\_gsi\_sysconfig\_get\_proc\_id\_string\_unix ( char \*\* *proc\_id\_string* )

Get a unique string representing the current process.

This is just the pid converted to a string.

#### Parameters

<i>proc_id_string</i>	A unique string representing the process
-----------------------	--

#### Returns

GLOBUS\_SUCCESS unless an error occurred

#### 3.2.2.5 globus\_result\_t globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_unix ( char \* *filename*, char \*\* *absolute\_path* )

Make the filename into an absolute path string based on the current working directory.

#### Parameters

<i>filename</i>	the filename to get the absolute path of.
<i>absolute_path</i>	The resulting absolute path. This needs to be freed when no longer needed.

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.6 globus\_result\_t globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_unix ( char \* *full\_filename*, char \*\* *dir\_string*, char \*\* *filename\_string* )

Split the directory and filename portions of a filename string into two separate strings.

#### Parameters

<i>full_filename</i>	The filename to split. Splits on the last occurrence of '/' where the directory is everything before the last '/', and the filename is everything after.
<i>dir_string</i>	The directory portion of the filename string. If no '/' is found throughout the string, this variable points to NULL. This needs to be freed when no longer needed.
<i>filename_string</i>	The filename portion of the filename string. If no '/' is found throughout, this variable is a duplicate of the full_filename parameter. This needs to be freed when no longer needed.

#### Returns

GLOBUS\_SUCCESS if no error occurred. Otherwise an error object ID is returned.

#### 3.2.2.7 globus\_result\_t globus\_gsi\_sysconfig\_get\_current\_working\_dir\_unix ( char \*\* *working\_dir* )

Get the current working directory on the system.

#### Parameters

<i>working_dir</i>	The current working directory
--------------------	-------------------------------

#### Returns

GLOBUS\_SUCCESS or an error object identifier

#### 3.2.2.8 globus\_result\_t globus\_gsi\_sysconfig\_get\_home\_dir\_unix ( char \*\* *home\_dir* )

Get the HOME Directory of the current user.

Should be the \$HOME environment variable.

#### Parameters

<i>home_dir</i>	The home directory of the current user
-----------------	--

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object is returned.

#### 3.2.2.9 globus\_result\_t globus\_gsi\_sysconfig\_file\_exists\_unix ( const char \* *filename* )

Check if the file exists.

#### Parameters

<i>filename</i>	The filename of the file to check for
-----------------	---------------------------------------

#### Returns

GLOBUS\_SUCCESS if the file exists and is readable, otherwise an error object identifier

#### 3.2.2.10 globus\_result\_t globus\_gsi\_sysconfig\_dir\_exists\_unix ( const char \* *filename* )

Check if the directory exists.

#### Parameters

<i>filename</i>	The filename of the directory to check for
-----------------	--

#### Returns

GLOBUS\_SUCCESS if the directory exists, otherwise an error object identifier.

#### 3.2.2.11 globus\_result\_t globus\_gsi\_sysconfig\_check\_keyfile\_unix ( const char \* *filename* )

This is a convenience function used to check the status of a private key file.

The desired status is only the current user has ownership and read permissions, everyone else should not be able to access it.

#### Parameters

<i>filename</i>	The name of the file to check the status of
-----------------	---

## Returns

GLOBUS\_SUCCESS if the status of the file was able to be determined. Otherwise, an error object identifier

### 3.2.2.12 globus\_result\_t globus\_gsi\_sysconfig\_check\_certfile\_unix ( const char \* filename )

This is a convenience function used to check the status of a certificate file.

The desired status is the current user has ownership and read/write permissions, while group and others only have read permissions.

## Parameters

<i>filename</i>	The name of the file to check the status of
-----------------	---

## Returns

GLOBUS\_SUCCESS if the status of the file was able to be determined. Otherwise, an error object identifier

### 3.2.2.13 globus\_result\_t globus\_gsi\_sysconfig\_get\_cert\_dir\_unix ( char \*\* cert\_dir )

Get the Trusted Certificate Directory containing the trusted Certificate Authority certificates.

This directory is determined in the order shown below. Failure in one method results in attempting the next.

1. **X509\_CERT\_DIR environment variable** - if this is set, the trusted certificates will be searched for in that directory. This variable allows the end user to specify the location of trusted certificates.
2. **\$HOME/.globus/certificates** - If this directory exists, and the previous methods of determining the trusted certs directory failed, this directory will be used.
3. **/etc/grid-security/certificates** - This location is intended to be independent of the globus installation (\$GLOBUS\_LOCATION), and is generally only writeable by the host system administrator.
4. **\$GLOBUS\_LOCATION/share/certificates**

## Parameters

<i>cert_dir</i>	The trusted certificates directory
-----------------	------------------------------------

## Returns

GLOBUS\_SUCCESS if no error occurred, and a sufficient trusted certificates directory was found. Otherwise, an error object identifier returned.

### 3.2.2.14 globus\_result\_t globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_unix ( char \*\* user\_cert, char \*\* user\_key )

Get the User Certificate Filename based on the current user's environment.

The following locations are searched for cert and key files in order:

1. environment variables X509\_USER\_CERT and X509\_USER\_KEY
2. \$HOME/.globus/usercert.pem and \$HOME/.globus/userkey.pem
3. \$HOME/.globus/usercred.p12 - this is a PKCS12 credential

#### Parameters

<i>user_cert</i>	pointer the filename of the user certificate
<i>user_key</i>	pointer to the filename of the user key

#### Returns

GLOBUS\_SUCCESS if the cert and key files were found in one of the possible locations, otherwise an error object identifier is returned

#### 3.2.2.15 globus\_result\_t globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_unix ( char \*\* *host\_cert*, char \*\* *host\_key* )

Get the Host Certificate and Key Filenames based on the current user's environment.

The host cert and key are searched for in the following locations (in order):

1. X509\_USER\_CERT and X509\_USER\_KEY environment variables
2. registry keys x509\_user\_cert and x509\_user\_key in software\Globus\GSI
3. \<GLOBUS\_LOCATION\>\etc\host[cert|key].pem
4. \<users home directory\>\.globus\host[cert|key].pem

#### Parameters

<i>host_cert</i>	pointer to the host certificate filename
<i>host_key</i>	pointer to the host key filename

#### Returns

GLOBUS\_SUCCESS if the host cert and key were found, otherwise an error object identifier is returned

#### 3.2.2.16 globus\_result\_t globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_unix ( char \* *service\_name*, char \*\* *service\_cert*, char \*\* *service\_key* )

Get the Service Certificate Filename based on the current user's environment.

The host cert and key are searched for in the following locations (in order):

1. X509\_USER\_CERT and X509\_USER\_KEY environment variables
2. \etc\grid-security\{service\_name}\{service\_name}[cert|key].pem
3. GLOBUS\_LOCATION\etc\{service\_name}\{service\_name}[cert|key].pem So for example, if my service was named: myservice, the location of the certificate would be: GLOBUS\_LOCATION\etc\myservice\myservicecert.pem
4. \<users home\>\.globus\{service\_name}\{service\_name}[cert|key].pem

#### Parameters

<i>service_name</i>	The name of the service which allows us to determine the locations of cert and key files to look for
<i>service_cert</i>	pointer to the host certificate filename
<i>service_key</i>	pointer to the host key filename

## Returns

GLOBUS\_SUCCESS if the service cert and key were found, otherwise an error object identifier

**3.2.2.17** `globus_result_t globus_gsi_sysconfig_get_proxy_filename_unix ( char ** user_proxy, globus_gsi_proxy_file_type_t proxy_file_type )`

Get the proxy cert filename based on the following search order:

1. X509\_USER\_PROXY environment variable - This environment variable is set by the at run time for the specific application. If the proxy\_file\_type variable is set to GLOBUS\_PROXY\_OUTPUT (a proxy filename for writing is requested), and the X509\_USER\_PROXY is set, this will be the resulting value of the user\_proxy filename string passed in. If the proxy\_file\_type is set to GLOBUS\_PROXY\_INPUT and X509\_USER\_PROXY is set, but the file it points to does not exist, or has some other readability issues, the function will continue checking using the other methods available.
2. Check the default location for the proxy file of `\\tmp\\x509_u\\<user_id\\>` where `\\<user id\\>` is some unique string for that user on the host

## Parameters

<i>user_proxy</i>	the proxy filename of the user
<i>proxy_file_type</i>	Switch for determining whether to return a existing proxy filename or if a filename suitable for creating a proxy should be returned

## Returns

GLOBUS\_SUCCESS or an error object identifier

**3.2.2.18** `globus_result_t globus_gsi_sysconfig_get_signing_policy_filename_unix ( X509_NAME * ca_name, char * cert_dir, char ** signing_policy_filename )`

Get the Signing Policy Filename on the current system, based on the CA's subject name, and the trusted certificates directory.

## Parameters

<i>ca_name</i>	The X509 subject name of the CA to get the signing policy of. The hash of the CA is generated from this
<i>cert_dir</i>	The trusted CA certificates directory, containing the singing_policy files of the trusted CA's.
<i>signing_policy_filename</i>	The resulting singing_policy filename

## Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID

**3.2.2.19** `globus_result_t globus_gsi_sysconfig_get_ca_cert_files_unix ( char * ca_cert_dir, globus_fifo_t * ca_cert_list )`

Gets a list of trusted CA certificate filenames in a trusted CA certificate directory.

## Parameters

<i>ca_cert_dir</i>	The trusted CA certificate directory to get the filenames from
<i>ca_cert_list</i>	The resulting list of CA certificate filenames. This is a a globus list structure.

See also

globus\_fifo\_t

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.20 globus\_result\_t globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_unix ( char \* *default\_filename* )

Removes all proxies (ie.

all delegated and grid-proxy-init generated proxies) found in the secure tmp directory that are owned by the current user.

#### Parameters

<i>default_filename</i>	The filename of the default proxy
-------------------------	-----------------------------------

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.21 globus\_result\_t globus\_gsi\_sysconfig\_is\_superuser\_unix ( int \* *is\_superuser* )

Checks whether the current user is root.

#### Parameters

<i>is_superuser</i>	1 if the user is the superuser 0 if not
---------------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.22 globus\_result\_t globus\_gsi\_sysconfig\_get\_gridmap\_filename\_unix ( char \*\* *filename* )

Get the path and file name of the grid map file.

#### Parameters

<i>filename</i>	Contains the location of the grid map file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.23 globus\_result\_t globus\_gsi\_sysconfig\_get\_authz\_conf\_filename\_unix ( char \*\* *filename* )

Get the path and file name of the authorization callback configuration file.

#### Parameters

<i>filename</i>	Contains the location of the authorization callback configuration file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.24 globus\_result\_t globus\_gsi\_sysconfig\_get\_authz\_lib\_conf\_filename\_unix ( char \*\* filename )

Get the path and file name of the authorization callback configuration file.

#### Parameters

<i>filename</i>	Contains the location of the authorization callback configuration file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.2.2.25 globus\_result\_t globus\_gsi\_sysconfig\_get\_gaa\_conf\_filename\_unix ( char \*\* filename )

Get the path and file name of the GAA configuration file.

#### Parameters

<i>filename</i>	Contains the location of the GAA callback configuration file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned



### 3.3 Functions for Win32 platforms

#### Win32 - Set Key Permissions

- globus\_result\_t **globus\_gsi\_sysconfig\_set\_key\_permissions\_win32** (char \*filename)

#### Win32 - File Exists

- globus\_result\_t **globus\_gsi\_sysconfig\_file\_exists\_win32** (const char \*filename)

#### Win32 - Directory Exists

- globus\_result\_t **globus\_gsi\_sysconfig\_dir\_exists\_win32** (const char \*filename)

#### Win32 - Get Current Working Directory

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_current\_working\_dir\_win32** (char \*\*working\_dir)

#### Win32 - Make Absolute Path

- globus\_result\_t **globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_win32** (char \*filename, char \*\*absolute\_path)

#### Win32 - Split Directory and Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_win32** (char \*full\_filename, char \*\*dir\_string, char \*\*filename\_string)

#### Win32 - Get Trusted CA Cert Dir

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_cert\_dir\_win32** (char \*\*cert\_dir)

#### Win32 - Get User Certificate Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_win32** (char \*\*user\_cert, char \*\*user\_key)

#### Win32 - Get Host Certificate and Key Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_win32** (char \*\*host\_cert, char \*\*host\_key)

#### Win32 - Get Service Certificate and Key Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_win32** (char \*service\_name, char \*\*service\_cert, char \*\*service\_key)

### Win32 - Get Proxy Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_proxy\_filename\_win32** (char \*\*user\_proxy, **globus\_gsi\_proxy\_file\_type\_t** proxy\_file\_type)

### Win32 - Get CA Cert Filenames

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_ca\_cert\_files\_win32** (char \*ca\_cert\_dir, globus\_fifo\_t \*ca\_cert\_list)

### Win32 - Remove all proxies owned by current uid

- globus\_result\_t **globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_win32** (char \*default\_filename)

### Win32 - Get the path and file name of the grid map file

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_gridmap\_filename\_win32** (char \*\*filename)
- globus\_result\_t **globus\_gsi\_sysconfig\_get\_authz\_conf\_filename\_win32** (char \*\*filename)

### Win32 - Get the path and file name of the gaa config file

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_gaa\_conf\_filename\_win32** (char \*\*filename)

### Win32 - Check if the current user is root

- globus\_result\_t **globus\_gsi\_sysconfig\_is\_superuser\_win32** (int \*is\_superuser)

### Win32 - Get Signing Policy Filename

- globus\_result\_t **globus\_gsi\_sysconfig\_get\_signing\_policy\_filename\_win32** (X509\_NAME \*ca\_name, char \*cert\_dir, char \*\*signing\_policy\_filename)

### 3.3.1 Detailed Description

These functions implement the Win32 version of the Globus GSI System Configuration API. **They should never be called directly, please use the provided platform independent defines.**

### 3.3.2 Function Documentation

#### 3.3.2.1 globus\_result\_t globus\_gsi\_sysconfig\_set\_key\_permissions\_win32 ( char \* filename )

Set the file permissions of a file to read only by the user which are the permissions that should be set for all private keys.

#### Parameters

<i>filename</i>	
-----------------	--

#### Returns

GLOBUS\_SUCCESS or an error object id

#### 3.3.2.2 globus\_result\_t globus\_gsi\_sysconfig\_file\_exists\_win32 ( const char \* *filename* )

Check that the file exists.

#### Parameters

<i>filename</i>	the file to check
-----------------	-------------------

#### Returns

GLOBUS\_SUCCESS (even if the file doesn't exist) - in some abortive cases an error object identifier is returned

#### 3.3.2.3 globus\_result\_t globus\_gsi\_sysconfig\_dir\_exists\_win32 ( const char \* *filename* )

Check that the directory exists.

#### Parameters

<i>filename</i>	the file to check
-----------------	-------------------

#### Returns

GLOBUS\_SUCCESS if the directory exists, otherwise an error object identifier.

#### 3.3.2.4 globus\_result\_t globus\_gsi\_sysconfig\_get\_current\_working\_dir\_win32 ( char \*\* *working\_dir* )

Get the current working directory on a windows system.

#### Parameters

<i>working_dir</i>	The working directory to get
--------------------	------------------------------

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.3.2.5 globus\_result\_t globus\_gsi\_sysconfig\_make\_absolute\_path\_for\_filename\_win32 ( char \* *filename*, char \*\* *absolute\_path* )

Make the filename into an absolute path string based on the current working directory.

#### Parameters

<i>filename</i>	the filename to get the absolute path of.
<i>absolute_path</i>	The resulting absolute path

## Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

**3.3.2.6 globus\_result\_t globus\_gsi\_sysconfig\_split\_dir\_and\_filename\_win32 ( char \* *full\_filename*, char \*\* *dir\_string*, char \*\* *filename\_string* )**

Split the directory and filename portions of a filename string into two separate strings.

## Parameters

<i>full_filename</i>	
<i>dir_string</i>	
<i>filename_string</i>	

## Returns

**3.3.2.7 globus\_result\_t globus\_gsi\_sysconfig\_get\_cert\_dir\_win32 ( char \*\* *cert\_dir* )**

Get the Trusted Certificate Directory containing the trusted Certificate Authority certificates.

This directory is determined in the order shown below. Failure in one method results in attempting the next.

1. **X509\_CERT\_DIR environment variable** - if this is set, the trusted certificates will be searched for in that directory. This variable allows the end user to specify the location of trusted certificates.
2. **x509\_cert\_dir registry key** - If this registry key is set on windows, the directory it points to should contain the trusted certificates. The path to the registry key is software\Globus\GSI
3. **\<user home directory>\.globus\certificates** - If this directory exists, and the previous methods of determining the trusted certs directory failed, this directory will be used.
4. **Host Trusted Cert Dir** - This location is intended to be independent of the globus installation (\$GLOBUS\_LOCATION), and is generally only writeable by the host system administrator.
5. **Globus Install Trusted Cert Dir** - this is \$GLOBUS\_LOCATION\share\certificates.

## Parameters

<i>cert_dir</i>	The trusted certificates directory
-----------------	------------------------------------

## Returns

GLOBUS\_SUCCESS if no error occurred, and a sufficient trusted certificates directory was found. Otherwise, an error object identifier returned.

**3.3.2.8 globus\_result\_t globus\_gsi\_sysconfig\_get\_user\_cert\_filename\_win32 ( char \*\* *user\_cert*, char \*\* *user\_key* )**

Get the User Certificate Filename based on the current user's environment.

The following locations are searched for cert and key files in order:

1. environment variables X509\_USER\_CERT and X509\_USER\_KEY
2. registry keys x509\_user\_cert and x509\_user\_key in software\Globus\GSI

3. <users home directory>\.globus\usercert.pem and <users home directory>\.globus\userkey.pem
4. <users home directory>\.globus\usercred.p12 - this is a PKCS12 credential

#### Parameters

<i>user_cert</i>	pointer the filename of the user certificate
<i>user_key</i>	pointer to the filename of the user key

#### Returns

GLOBUS\_SUCCESS if the cert and key files were found in one of the possible locations, otherwise an error object identifier is returned

#### 3.3.2.9 globus\_result\_t globus\_gsi\_sysconfig\_get\_host\_cert\_filename\_win32 ( char \*\* *host\_cert*, char \*\* *host\_key* )

Get the Host Certificate and Key Filenames based on the current user's environment.

The host cert and key are searched for in the following locations (in order):

1. X509\_USER\_CERT and X509\_USER\_KEY environment variables
2. registry keys x509\_user\_cert and x509\_user\_key in software\Globus\GSI
3. <GLOBUS\_LOCATION>\etc\host[cert|key].pem
4. <users home directory>\.globus\host[cert|key].pem

#### Parameters

<i>host_cert</i>	pointer to the host certificate filename
<i>host_key</i>	pointer to the host key filename

#### Returns

GLOBUS\_SUCCESS if the host cert and key were found, otherwise an error object identifier is returned

#### 3.3.2.10 globus\_result\_t globus\_gsi\_sysconfig\_get\_service\_cert\_filename\_win32 ( char \* *service\_name*, char \*\* *service\_cert*, char \*\* *service\_key* )

Get the Service Certificate Filename based on the current user's environment.

The host cert and key are searched for in the following locations (in order):

1. X509\_USER\_CERT and X509\_USER\_KEY environment variables
2. registry keys x509\_user\_cert and x509\_user\_key in software\Globus\GSI
3. GLOBUS\_LOCATION\etc\{service\_name}\{service\_name}[cert|key].pem So for example, if my service was named: myservice, the location of the certificate would be: GLOBUS\_LOCATION\etc\myservice\myservicecert.pem
4. <users home>\.globus\{service\_name}\{service\_name}[cert|key].pem

#### Parameters

<i>service_name</i>	The name of the service which allows us to determine the locations of cert and key files to look for
<i>service_cert</i>	pointer to the host certificate filename
<i>service_key</i>	pointer to the host key filename

#### Returns

GLOBUS\_SUCCESS if the service cert and key were found, otherwise an error object identifier

**3.3.2.11** `globus_result_t globus_gsi_sysconfig_get_proxy_filename_win32 ( char ** user_proxy, globus_gsi_proxy_file_type_t proxy_file_type )`

Get the proxy cert filename based on the following search order:

1. X509\_USER\_PROXY environment variable - This environment variable is set by the at run time for the specific application. If the proxy\_file\_type variable is set to GLOBUS\_PROXY\_OUTPUT (a proxy filename for writing is requested), and the X509\_USER\_PROXY is set, this will be the resulting value of the user\_proxy filename string passed in. If the proxy\_file\_type is set to GLOBUS\_PROXY\_INPUT and X509\_USER\_PROXY is set, but the file it points to does not exist, or has some other readability issues, the function will continue checking using the other methods available.
2. check the registry key: x509\_user\_proxy. Just as with the environment variable, if the registry key is set, and proxy\_file\_type is GLOBUS\_PROXY\_OUTPUT, the string set to be the proxy filename will be this registry key's value. If proxy\_file\_type is GLOBUS\_PROXY\_INPUT, and the file doesn't exist, the function will check the next method for the proxy's filename.
3. Check the default location for the proxy file. The default location should be set to reside in the temp directory on that host, with the filename taking the format: x509\_u<user id> where <user id> is some unique string for that user on the host

#### Parameters

<i>user_proxy</i>	the proxy filename of the user
<i>proxy_file_type</i>	Switch for determining whether to return a existing proxy filename or if a filename suitable for creating a proxy should be returned

#### Returns

GLOBUS\_SUCCESS or an error object identifier

**3.3.2.12** `globus_result_t globus_gsi_sysconfig_get_ca_cert_files_win32 ( char * ca_cert_dir, globus_fifo_t * ca_cert_list )`

Gets a list of trusted CA certificate filenames in a trusted CA certificate directory.

#### Parameters

<i>ca_cert_dir</i>	The trusted CA certificate directory to get the filenames from
<i>ca_cert_list</i>	The resulting list of CA certificate filenames. This is a a globus list structure.

#### See also

globus\_fifo\_t

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.3.2.13 globus\_result\_t globus\_gsi\_sysconfig\_remove\_all\_owned\_files\_win32 ( char \* *default\_filename* )

Removes all proxies (ie.

all delegated and grid-proxy-init generated proxies) found in the secure tmp directory that are owned by the current user.

#### Parameters

<i>default_filename</i>	The filename of the default proxy
-------------------------	-----------------------------------

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.3.2.14 globus\_result\_t globus\_gsi\_sysconfig\_get\_gridmap\_filename\_win32 ( char \*\* *filename* )

Get the path and file name of the grid map file.

#### Parameters

<i>filename</i>	Contains the location of the grid map file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.3.2.15 globus\_result\_t globus\_gsi\_sysconfig\_get\_authz\_conf\_filename\_win32 ( char \*\* *filename* )

Get the path and file name of the authorization callback configuration file.

#### Parameters

<i>filename</i>	Contains the location of the authorization callback configuration file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

#### 3.3.2.16 globus\_result\_t globus\_gsi\_sysconfig\_get\_gaa\_conf\_filename\_win32 ( char \*\* *filename* )

Get the path and file name of the gaa config configuration file .

#### Parameters

<i>filename</i>	Contains the location of the authorization callback configuration file upon successful return
-----------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

### 3.3.2.17 globus\_result\_t globus\_gsi\_sysconfig\_is\_superuser\_win32 ( int \* *is\_superuser* )

Checks whether the current user is root.

#### Parameters

<i>is_superuser</i>	1 if the user is the superuser 0 if not
---------------------	---

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID is returned

### 3.3.2.18 globus\_result\_t globus\_gsi\_sysconfig\_get\_signing\_policy\_filename\_win32 ( X509\_NAME \* *ca\_name*, char \* *cert\_dir*, char \*\* *signing\_policy\_filename* )

Get the Signing Policy Filename on the current system, based on the CA's subject name, and the trusted certificates directory.

#### Parameters

<i>ca_name</i>	The X509 subject name of the CA to get the signing policy of. The hash of the CA is generated from this
<i>cert_dir</i>	The trusted CA certificates directory, containing the signing_policy files of the trusted CA's.
<i>signing_policy_filename</i>	The resulting signing_policy filename

#### Returns

GLOBUS\_SUCCESS if no error occurred, otherwise an error object ID



## 3.4 Functions for all platforms

### Get Unique Proxy Filename

- `globus_result_t globus_gsi_sysconfig_get_unique_proxy_filename (char **unique_filename)`

#### 3.4.1 Detailed Description

These functions are platform independent members of the Globus GSI System Configuration API.

#### 3.4.2 Function Documentation

##### 3.4.2.1 `globus_result_t globus_gsi_sysconfig_get_unique_proxy_filename ( char ** unique_filename )`

Get a unique proxy cert filename.

This is mostly used for delegated proxy credentials. Each filename returned is going to be unique for each time the function is called.

#### Parameters

<i>unique_filename</i>	the unique filename for a delegated proxy cert
------------------------	--

#### Returns

GLOBUS\_SUCCESS or an error object identifier

## 3.5 Activation

### Defines

- **#define GLOBUS\_GSI\_SYSCONFIG\_MODULE**

#### 3.5.1 Detailed Description

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

```
globus_module_activate(GLOBUS_GSI_SYSCONFIG_MODULE)
```

This function returns GLOBUS\_SUCCESS if the Globus GSI System Configuration API was successfully initialized, and you are therefore allowed to subsequently call Globus GSI System Configuration 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 GSI System Configuration API, the following function must be called:

```
globus_module_deactivate(GLOBUS_GSI_SYSCONFIG_MODULE)
```

This function should be called once for each time Globus GSI System Configuration API was activated.

#### 3.5.2 Define Documentation

##### 3.5.2.1 #define GLOBUS\_GSI\_SYSCONFIG\_MODULE

Module descriptor.

## 3.6 Datatypes

### Enumerations

- enum **globus\_gsi\_sysconfig\_error\_t** { **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_SUCCESS** = 0, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_DIR** = 1, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_STRING** = 2, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_KEY\_STRING** = 3, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_HOME\_DIR** = 4, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_ERRNO** = 5, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_CHECKING\_FILE\_EXISTS** = 6, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_FILENAME** = 7, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_PROXY\_FILENAME** = 8, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_DELEG\_FILENAME** = 9, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CA\_CERT\_FILENAMES** = 10, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CWD** = 11, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_REMOVING\_OWNED\_FILES** = 12, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_GRIDMAP\_FILENAME** = 13, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_CHECKING\_SUPERUSER** = 14, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_SETTING\_PERMS** = 15, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_SIGNING\_POLICY** = 16, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_PW\_ENTRY** = 17, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_AUTHZ\_FILENAME** = 18, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_REGULAR** = 19, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_DOES\_NOT\_EXIST** = 20, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_BAD\_PERMISSIONS** = 21, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_OWNED** = 22, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_IS\_DIR** = 23, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_ZERO\_LENGTH** = 24, **GLOBUS\_GSI\_SYSCONFIG\_INVALID\_ARG** = 25, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_HAS\_LINKS** = 26, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_HAS\_CHANGED** = 27, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_AUTHZ\_LIB\_FILENAME** = 28, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_GAA\_FILENAME** = 29, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_DIR** = 30, **GLOBUS\_GSI\_SYSCONFIG\_ERROR\_LAST** = 31 }
- enum **globus\_gsi\_proxy\_file\_type\_t** { **GLOBUS\_PROXY\_FILE\_INPUT**, **GLOBUS\_PROXY\_FILE\_OUTPUT** }

### 3.6.1 Enumeration Type Documentation

#### 3.6.1.1 enum **globus\_gsi\_sysconfig\_error\_t**

GSI System Config Error codes.

Enumerator:

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_SUCCESS** Success - never used.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_DIR** Unable to determine trusted certificates directory.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_STRING** Error while generating certificate filename.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_KEY\_STRING** Error while generating private key filename.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_HOME\_DIR** Unable to determine user's home directory.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_ERRNO** System Error -- see underlying error for details.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_CHECKING\_FILE\_EXISTS** Unable to determine whether file exists.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CERT\_FILENAME** Unable to determine the location of the certificate file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_PROXY\_FILENAME** Unable to determine the location of the proxy file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_DELEG\_FILENAME** Unable to determine the location of the delegated proxy file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CA\_CERT\_FILENAMES** Unable to generate a list of CA certificate filenames.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_CWD** Error while discovering the current working directory.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_REMOVING\_OWNED\_FILES** Failed to remove all proxy files.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_GRIDMAP\_FILENAME** Unable to determine the location of the grid map file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_CHECKING\_SUPERUSER** Failure while checking whether the current user is the super user.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_SETTING\_PERMS** Error while trying to set file permissions.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_SIGNING\_POLICY** Unable to determine the location of a signing policy file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_PW\_ENTRY** Could not find password entry for user.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_AUTHZ\_FILENAME** Failed to locate the authorization callout configuration file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_REGULAR** File is not a regular file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_DOES\_NOT\_EXIST** File does not exist.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_BAD\_PERMISSIONS** File has incorrect permissions for operation.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_OWNED** File is not owned by current user.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_IS\_DIR** File is a directory.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_ZERO\_LENGTH** File has zero length.

**GLOBUS\_GSI\_SYSCONFIG\_INVALID\_ARG** Invalid argument.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_HAS\_LINKS** File has more than one link.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_HAS\_CHANGED** File has changed in the meantime.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_AUTHZ\_LIB\_FILENAME** Failed to locate the authorization callout library configuration file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_GETTING\_GAA\_FILENAME** Failed to locate the gaa configuration file.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_FILE\_NOT\_DIR** File is not a directory.

**GLOBUS\_GSI\_SYSCONFIG\_ERROR\_LAST** Last marker - never used.

### 3.6.1.2 enum globus\_gsi\_proxy\_file\_type\_t

Enumerator used to keep track of input/output types of filenames.

Enumerator:

**GLOBUS\_PROXY\_FILE\_INPUT** The proxy filename is intended for reading (it should already exist)

**GLOBUS\_PROXY\_FILE\_OUTPUT** The proxy filename is intended for writing (it does not need to exist)