

GT4 Delegation Service Developer's Guide

DRAFT

GT4 Delegation Service Developer's Guide

Introduction

The Delegation Service provides an interface that, given the endpoint reference to the credential resource, allows service developers to retrieve a delegated credential. While there is a remote interface to delegate and refresh a credential, no remote interface is provided for acquiring the delegated credential (that is, all access is through a shared Java state.)

In addition, the component provides a utility API that can be used for developing client side code to generate a credential, delegate it and refresh it.

DRAFT

Table of Contents

1. Before you begin	1
1. Feature summary	1
2. Tested platforms	1
3. Backward compatibility summary	1
4. Technology dependencies	1
5. Security considerations	2
2. Developer Usage Scenarios	3
1. Java Usage Scenarios	3
2. C Usage Scenarios	5
3. Tutorials	7
4. Architecture and design overview	8
1. Overview	8
2. Relationship to WS-Trust	8
3. Normal Usage Patterns	9
4. Credential Storage	10
5. Access to Delegated Proxy Resources	10
6. Registration for Renewal Events	11
5. APIs	12
1. Programming Model Overview	12
2. Component API	12
6. Services and WSDL	13
1. Protocol overview	13
2. Operations	13
3. Delegation Service Resource properties	13
4. Faults	13
5. WSDL and Schema Definition	13
I. Command-line tools	?
globus-credential-delegate	15
globus-credential-refresh	16
globus-delegation-client	18
wsrf-destroy	21
wsrf-query	23
7. Configuring	25
1. Configuration overview	25
2. Syntax of the interface	25
8. Environment variable interface	27
1. Environmental variables for Message/Transport-level Security	27
9. Debugging	28
1. Logging in Java WS Core	28
2. Enabling verbose logging	29
3. Debugging info from delegation clients	29
10. Troubleshooting	30
1. Error Messages	31
2. CoG Configuration and troubleshooting	33
11. Related Documentation	34
Glossary	35

List of Figures

4.1. Delegation Service Creation	9
4.2. Delegation Service Refresh	10

DRAFT

List of Tables

1. globus-credential-delegate options	15
2. globus-credential-refresh options	17
3. Common options	19
4. Application-specific options	19
5. Common options	22
6. Common options	24
10.1. WS A&A Delegation Service Error Messages	32

DRAFT

Chapter 1. Before you begin

1. Feature summary

Features new in GT 4.2.0:

- Added support for GetResourceProperties and QueryResourceProperties interface.

Other Supported Features:

- Provides an interface for the delegation and renewal of credentials to a host.
- Allows for a single delegated credential to be reused across multiple service invocations (e.g. GRAM jobs).

Deprecated Features:

- None.

2. Tested platforms

Tested Platforms for Delegation Service:

- Windows XP
- Linux (Red Hat 7.3)

Tested Containers for Delegation Service:

- Java WS Core container
- Tomcat 5.0.30

3. Backward compatibility summary

Delegation Service has been updated to use the latest version of Java WS Core, which now supports the final version of WSRF/WSN specification¹. This service is not compatible with the previous stable versions, GT 4.0.x.

The Java WS Authentication and Message component has been updated to support RFC 3820 proxies by default as described here. This implies that credentials delegated in the previous stable version are incompatible with this version.

4. Technology dependencies

The Delegation Service depends on the following GT components:

- WS Authentication and Authorization
- Java WS Core

¹ <http://www.globus.org/toolkit/docs/4.2/4.2.0/common/javawscore/rn/index.html#id2534919>

The Delegation Service depends on all third party software Java WS Core² depends on.

5. Security considerations

5.1. Delegation Service Security Considerations

5.1.1. Key Pair Reuse

The current design re-uses the keys associated with the Delegation Service for each of the *proxy certificates* delegated to it. During a security review, it was pointed out that while this was fine from a cryptographic standpoint, compromising this single long-lived key pair may significantly extend the time for which a single intrusion (presuming an exploitable security flaw making the intrusion possible) is effective.

This can be remedied by either frequently regenerating the key pair used by the Delegation Service, which can be accomplished with a simple cron job, or by generating a new key pair for each new delegation. The latter of these approaches requires changes to the design and may be adopted in future versions of the toolkit. For the time being, we recommend the former approach should this issue concern you.

5.1.2. Authorizing Server prior to delegation

The delegation client that is distributed with the toolkit allows for delegation of credentials even when no authorization of the server is done. Also, when using secure message authentication, the authorization of the server is done after the completion of the operation. These two scenarios could lead to the delegation of credentials to a malicious server.

To prevent this, users should use secure *transport* (HTTPS) or GSI Secure Conversation and appropriate client-side authorization.

² <http://www.globus.org/toolkit/docs/4.2/4.2.0/common/javawscore/developer/javawscore-developer-beforeyoubegin.html#javawscore-tech-dependencies>

Chapter 2. Developer Usage Scenarios

1. Java Usage Scenarios

The `org.globus.delegation.DelegationUtil` provides an API that allows users to get the certificate chain resource property exposed by the Delegation Factory Service, delegate to a service, and to refresh and register listeners.

1.1. Client-side scenario

1.1.1. Getting a certificate chain of Delegation Factory Service

Prior to delegating, the client needs to get information about the public key of the Delegation Factory Service.

```
static X509Certificate[] getCertificateChainRP(String delegationUrl)
```

This takes an endpoint URL to a Delegation Factory Service and queries the `CertificateChain` resource property. The chain of certificates is returned as an array of `X509Certificate`. The client needs to delegate on the first certificate in the returned chain.

1.1.2. Delegating the credential

Once the delegation client has the public key of the Delegation Factory Service, it needs to create a delegated credential using that public key and then invoke the remote interface on the factory to delegate the credential. A utility API to do all of the above is described in [APIs](#).

```
public static EndpointReferenceType delegate(String delegationServiceUrl,
      GlobusCredential issuingCred,
      X509Certificate certificate,
      int lifetime,
      boolean fullDelegation,
      ClientSecurityDescriptor desc)
```

This utility method is used to create the security token to delegate using the `issuingCred` and `certificate` parameters. The lifetime and type of the delegated credential created is determined by the `lifetime` and `fullDelegation` parameters. The security token (delegated credential) thus created is then stored by the Delegation Factory Service specified by the `delegationServiceUrl`. The client security descriptor determines the authentication mechanism, protection and authorization settings to use.

The Endpoint Reference that is returned points to the delegated credential and can be used by co-hosted services (services in the same hosting environment) to retrieve the delegated credential.

1.1.3. Delegated Credential Lifetime

The lifetime of the delegated credential is set to be the lifetime of the WSRF resource created for the credential. This value is exposed as a resource property, `TerminationTime`, described in the WS-Lifetime specification.

The command line client `wsrf-get-property`¹ with the termination time resource property QName `{http://docs.oasis-open.org/wsr/2004/06/wsr-WS-ResourceLifetime-1.2-draft-01.xsd}TerminationTime` can be used to get the value for a given delegated credential.

¹ <http://common.javaworld.com/wsr/2004/06/wsr-get-property.html>

To obtain the termination time programmatically, the method `GetResourceProperty` with the above-mentioned `QName` can be used.

1.1.4. Refreshing the delegated credential

The user may need to refresh the delegated credential. The onus is on the user to do this prior to expiration of the delegated credential. If the user does not refresh the credential, the expired credential will be garbage collected and the Endpoint Reference cannot be reused.

```
public static void refresh(GlobusCredential issuingCred,
    X509Certificate certToSign,
    int lifetime,
    boolean fullDelegation,
    ClientSecurityDescriptor desc,
    EndpointReferenceType epr)
```

This method can be used to refresh a delegated credential that is referred to by the EPR *epr*. A new delegated credential is created using the *issuingCred*, *certToSign*, *lifetime* and *fullDelegation* parameters. The client security descriptor determines the authentication mechanism, protection and authorization type to use.

1.2. Service-side scenario

1.2.1. Registering a listener

This section describes the usage scenario where a service is provided with a delegated credential EPR and needs to access the credential. Typically, as a part of an application the delegated credential EPR is sent to the service and and it is assumed that the delegation service is co-hosted (that is, it runs in the same hosting environment).

The service needs to create a listener object that implements the `org.globus.delegation.DelegationRefreshListener` interface and register the listener with the Delegation Service. Upon registering the listener, the Delegation Service checks that the delegator identity matches either the identity passed in the subject object or the identity contained in the peer subject associated with the current message context. Once the listener has been authorized the delegated credential is made available to the listener.

```
static void
    registerDelegationListener(EndpointReferenceType epr,
    DelegationRefreshListener listener,
    Subject subject)
```

This method registers the listener *listener* with the delegation resource referenced by *epr*. The operation is permitted only if the identity in the subject object matches that of the user who delegated the credential.

```
static void
    registerDelegationListener(EndpointReferenceType epr,
    DelegationRefreshListener listener)
```

This method provides the same functionality as the previous one, except that the subject object is picked up from the property `org.globus.wsrfl.security.Constants.PEER_SUBJECT` in the current message context. If the identity of the user who delegated the credential matches that of the subject object referred to by the property, then the operation is permitted.

2. C Usage Scenarios

The `globus_c_delegation_client_util` package provides functions to create new delegation resources and refresh the credential owned by an existing delegation resource.

2.1. Module Activation

Activate the `GLOBUS_DELEGATION_CLIENT_UTIL_MODULE` before calling any functions in the `globus_c_delegation_client_util` API.

This fragment demonstrates activating the module. If successful, `rc` will be set to 0.

```
#include "globus_delegation_client_util.h"

int rc;

rc = globus_module_activate(GLOBUS_DELEGATION_CLIENT_UTIL_MODULE);
```

2.2. Create a Handle

A `globus_delegation_client_util_handle_t` structure manages the state needed to process the delegation or refresh operations. A process may create multiple `globus_delegation_client_util_handle_t` structures, but each may be processing only one delegation or refresh operation at a time.

The function `globus_delegation_client_util_handle_init` creates a new handle. This fragment demonstrates creating a new handle with default parameters.

```
globus_delegation_client_util_handle_t handle;
globus_soap_message_attr_t message_attr = NULL;
globus_handler_chain_t handler_chain = NULL;
const globus_gsi_cred_handle_t user_cred = NULL;
globus_delegation_client_util_attr_t attr = NULL;
globus_result_t result;

/* Initialize attributes, user_cred, handler chain if needed */

result = globus_delegation_client_util_handle_init(
&handle,
attr,
handler_chain,
user_cred,
attr)
```

2.3. Delegating a credential

The `globus_c_delegation_client_util` API provides four different functions for delegating credentials. They differ in two facets. One is whether the function is blocking or nonblocking; the other is whether the function contacts a resource based on a service URL or a complete endpoint reference. Nonblocking functions have names which end with `_register` and are passed a callback function pointer which will be called when the delegation completes. Functions which use Endpoint References have `_epr` in their names.

This fragment demonstrates delegating a credential using the blocking interface taking a service URL. If successful, result will equal GLOBUS_SUCCESS and delegated_epr will point to the endpoint reference naming the delegated credential.

```
globus_result_t          result;
    wsa_EndpointReferenceType *    delegated_epr = NULL;

    result = globus_delegation_client_util_delegate(
        handle,
        "http://my.virtual.org:8443/wsrf/services/DelegationFactoryService",
        GLOBUS_FALSE,
        &delegated_epr);
```

2.4. Refreshing a credential

The globus_c_delegation_client_util API provides four different functions for refreshing credentials. They differ in two facets. One is whether the function is blocking or nonblocking; the other is whether the function contacts a resource based on a service URL or a complete endpoint reference. Nonblocking functions have names which end with *_register* and are passed a callback function pointer which will be called when the delegation completes. Functions which use Endpoint References have *_epr* in their names.

This fragment demonstrates refreshing a credential using the blocking interface taking an endpoint reference. If successful, result will equal GLOBUS_SUCCESS.

```
result = globus_delegation_client_util_refresh_epr(
    handle,
    delegated_epr,
    GLOBUS_FALSE);
```

Chapter 3. Tutorials

There are no tutorials available at this time.

DRAFT

Chapter 4. Architecture and design overview

1. Overview

This component offers an interface for delegating credentials and subsequently managing them. It exposes the delegated credentials as a WS-Resource and allows authorized co-hosted services to access these credentials through a Java API. Furthermore, it gives clients the ability to refresh and manage the lifetime of their delegated credentials.

This component has a Delegation Factory Service and Delegation Service. A delegate call on the factory creates a WS-Resource managed by the Delegation Service that represents the delegated credential. The delegate call returns an Endpoint Reference (EPR) that can be used to later refresh the credentials.

Services that are interested in the delegated credential can register a listener (an object that implements `org.globus.delegation.DelegationRefreshListener`) with the specific delegated credential resource. There currently is no remote interface for this, hence only services that are in the same hosting environment can register interest. The credentials are pushed to the listener anytime a refresh is done.

In practice, delegation is done as follows:

1. The Delegation Factory Service publishes its certificate chain, including the service's certificate, as a Resource Property.
2. In the first step of the delegation process the client obtains this certificate chain, validates and authorizes it, and extracts the *public key* from the Delegation Factory *Service certificate*.
3. The client then creates the *proxy certificate* it is going to delegate by binding, i.e. signing, the service's public key to the proxy certificate information using its *private key*.
4. In the third and final step, the client passes the certificate chain that starts with the proxy certificate to the Delegation Factory Service, which upon receipt replies with the address to the created delegated credential WS-Resource.

Additionally, the component allows the delegator to renew delegated credentials. Credential renewal follows the same steps as the initial delegation except for the final step, in which the client acts against its WS-Resource instead of the Delegation Service Factory.

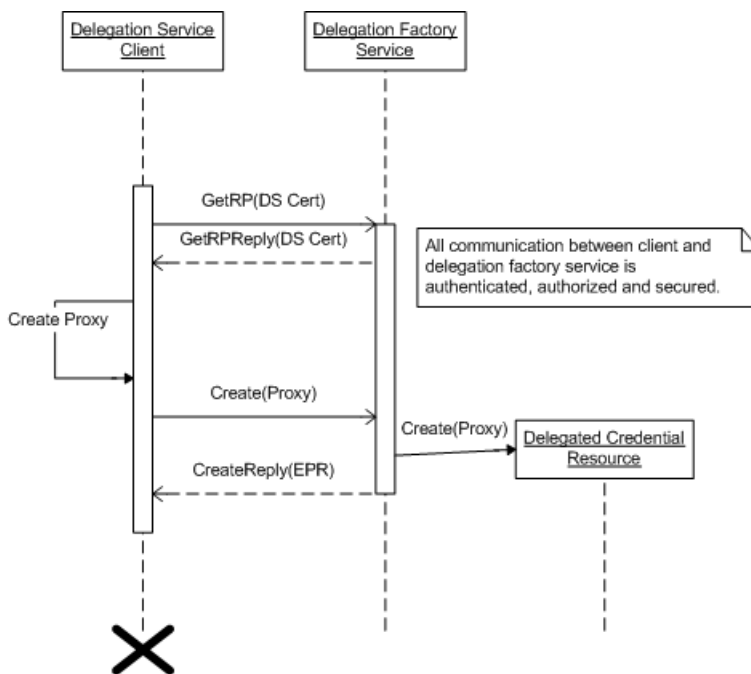
2. Relationship to WS-Trust

The Delegation Service uses WS-Trust messages as described in the WS-Trust specification. However, it should be noted that these messages are underspecified (the contents of the basic WS-Trust envelope are `xsd:any`) and the contents of these messages for the Delegation Services are simplistic and do not achieve the "spirit" intended by the specification.

3. Normal Usage Patterns

3.1. Creation

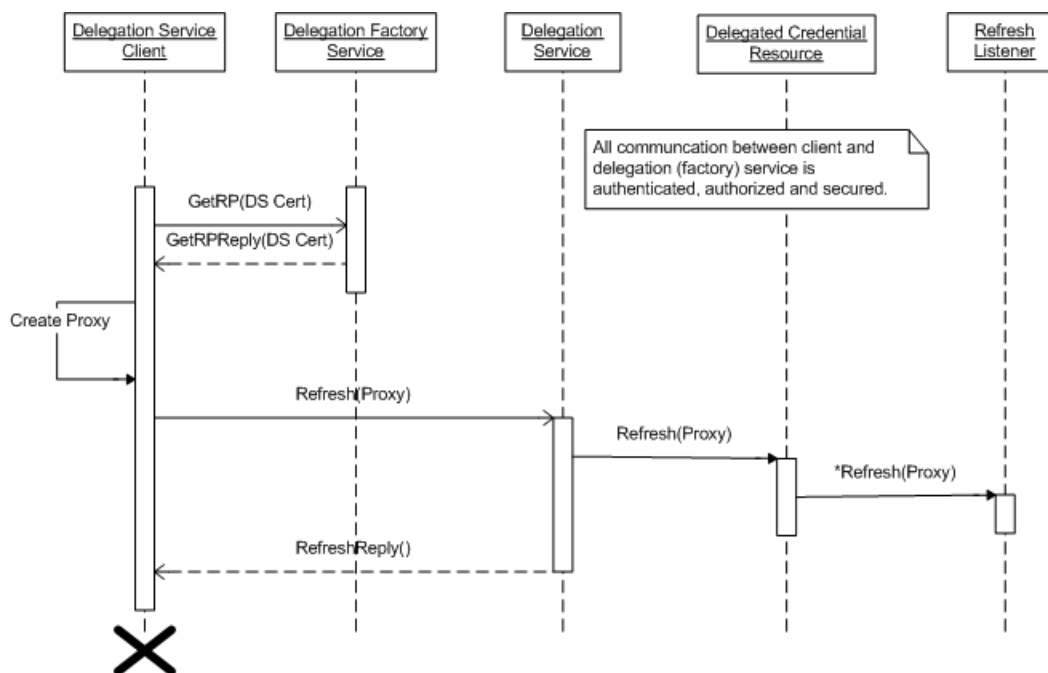
Figure 4.1. Delegation Service Creation



1. The client does a lookup on the Delegation Factory Service (DFS) for the Resource Property (RP) containing the certificate chain of the DFS. To perform this lookup, the client uses the EPR of the DFS obtained from an RP of another application (e.g. GRAM, RFT) or some other OOB method.
 - This lookup is secured via *transport-level security*. The client and server mutually authenticate and authorize each other. Messages are integrity-protected.
2. The client creates a proxy certificate by binding the public key of the DFS, obtained from the certificate chain in the previous step, to the proxy identity of the client.
 - The lifetime of a proxy certificate defaults to the lifetime of the signing certificate (typically a short-lived proxy).
3. The client sends the proxy certificate (and chain) to the DFS. An endpoint reference (EPR) for the delegated credential is returned to client, which the client may use in subsequent invocations of other services co-hosted with the Delegation Service.
 - The resource expires on proxy expiration.
 - An ACL is created for the delegated credential resource, which contains the identity of the credential delegator.

3.2. Refresh

Figure 4.2. Delegation Service Refresh



The original delegator of a credential may refresh the credential (i.e. replace it with a different credential, presumably one that has an expiration date farther in the future). The usage pattern for this method is identical to the Creation method described in the previous section, with the exception that the EPR of the previously created Delegation Resource is used as opposed to the EPR of the Delegation Factory Service.

4. Credential Storage

The Delegation Factory Service and the Delegation Service use standard Grid service credentials, namely a private key and certificate chain where the *EEC* has a DN containing the FQDN of the host on which the services are running. The private key and certificate chain are stored on the local disk, protected by local file system permissions. By default, the PEM files `/etc/grid-security/containercert.pem` and `/etc/grid-security/container-key.pem` are used.

When a delegated *proxy credential* is persisted to disk, it is stored as a serialized JAVA object in `~globus/.globus/persisted/{ip_addr}/DelegationResource/`. The private key of the Delegation Service is stored with the delegated proxy certificate to ease the use of delegated credentials by services in the hosting environment.

5. Access to Delegated Proxy Resources

After delegating a credential to the Delegation Service, a client will typically invoke an application service (e.g. GRAM, RFT) that requires the use of such a delegated credential. In such situations, the client will pass the EPR of the delegated credential to the service which it is invoking.

The service will use an internal hosting environment API (as opposed to a web services method) to access the delegated credential. This interface provides the identity of the requesting client to the underlying software, which verifies that the client identity is present in the ACL of the proxy resource before returning the requested credential.

6. Registration for Renewal Events

Services internal to the hosting environment can register with a resource proxy to receive updated credentials when they are renewed by the client. Such registrations are authorized in the same manner as direct access to the proxy. Registered services will have any renewed credentials pushed to them.

DRAFT

Chapter 5. APIs

1. Programming Model Overview

This component consists of two services: the Delegation Factory Service and the Delegation Service.

The Delegation Factory Service exposes its public certificate as a resource property and allows clients to delegate credentials bound to that *public key*. Upon delegation, an Endpoint Reference(EPR) to the delegated credential, which is implemented as a resource of the Delegation Service, is returned to the client. The client can use this EPR to provide a reference to the delegated credential to other services.

The Delegation Service itself has an interface to allow refreshing the credentials remotely. Other co-hosted services can register interest in delegated credentials through listeners and be notified when credentials are refreshed.

2. Component API

Some relevant API:

- `org.globus.delegation.DelegationUtil`
- `org.globus.delegation.DelegationRefreshListener`
- `org.globus.delegation.delegationService.DelegationPortType`
- `org.globus.delegation.delegationService.DelegationFactoryPortType`

Complete API:

- [Service API](#)¹
- [Common API](#)²

¹ http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_delegation_service_java/

² http://www.globus.org/api/javadoc-4.2.0/globus_wsrf_delegation_stubs_java/

Chapter 6. Services and WSDL

1. Protocol overview

The Delegation Service allows for delegation of credentials and is based on the [WS-Trust](#)¹ specification. A WSDL interface to refresh the credentials remotely is also provided. Access to these credentials is restricted to co-hosted services, i.e. services that are run in the same container, and is done using shared Java state. Co-hosted services interested in the credentials can register listeners and will be notified upon credential refresh.

2. Operations

2.1. Delegation Factory Service

- `RequestSecurityToken`: This operation allows for a security token to be sent to the service.

2.2. Delegation Service

- `refresh`: This operation is used to refresh a delegated credential. When invoked, all services that have registered interest in the credential through listeners are notified.

3. Delegation Service Resource properties

3.1. Delegation Factory Service

- `CertificateChain`: This resource property is used to expose the certificate used by delegation service.

4. Faults

All operations on Delegation Service and Delegation Factory Service throw `RemoteException` in case of failure.

5. WSDL and Schema Definition

- [Delegation Factory Service WSDL](#)²
- [Delegation Service WSDL](#)³

¹ <http://www.ibm.com/developerworks/library/ws-trust/>

² http://viewcvs.globus.org/viewcvs.cgi/ws-delegation/common/schema/delegationService/delegation_factory_flattened.wsdl?rev=1.3&only_with_tag=globus_4_2_0&content-type=text/vnd.viewcvs-markup

³ http://viewcvs.globus.org/viewcvs.cgi/ws-delegation/common/schema/delegationService/delegation_flattened.wsdl?rev=1.2&only_with_tag=globus_4_2_0&content-type=text/vnd.viewcvs-markup

Command-line tools

Note the **wsrf-destroy** and **wsrf-query** commands are common Java WS Core commands.

DRAFT

Name

globus-credential-delegate -- Delegation client

globus-credential-delegate

Tool description

Used to contact a Delegation Factory Service and store a delegated credential. A delegated credential is created and stored in a delegated credential WS-Resource, and the Endpoint Reference(EPR) of the credential is written out to a file for further use.

Command syntax

```
globus-credential-delegate [options] <eprFilename>
```

Table 1. globus-credential-delegate options

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-g, --delegation <mode>	Enables delegation. mode can be either ' limited ' or ' full '. Only supported with the GSI Secure Conversation authentication mechanism.
-help	Prints the usage message for the client.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-x, --proxyFilename <value>	Sets the proxy file to use as the client credential.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be ' msg ' for GSI Secure Message, or ' conv ' for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. type can be ' sig ' for signature or ' enc ' for encryption.
-s, --service <url>	Specifies the Delegation Factory Service URL.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-y, --lifetme <value>	Lifetime of delegated credential in seconds. Default is 43200 (which is 12 hours).
-z, --authorization <type>	Specifies authorization type. type can be ' self ', ' host ', ' none ', or a string specifying the expected identity of the remote party.
<eprFilename>	Filename to write the EPR of delegated credential to.

Name

`globus-credential-refresh -- Delegation refresh client`

`globus-credential-refresh`

Tool description

Used to refresh delegated credentials pointed to by the specified EPR. A new credential is generated and the one previously created by the Delegation Service is overwritten.

Command syntax

`globus-credential-refresh [options]`

DRAFT

Table 2. globus-credential-refresh options

-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-debug	Runs the client with debug message traces and error stack traces
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference. The EPR would be of the delegation resource that is to be refreshed.
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-help	Prints the usage message for the client.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while value is the simple value of the key. For complex keys, use the --eprFile option. For Delegation resource, the name will be as specified in the <i>delegationResourceKey</i> element and will replace <i>delegationResourceKey</i> with the actual key: -k " {http://www.globus.org/08/2004/delegationService}DelegationKey delegationResourceKey
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-s, --service <url>	Specifies the Delegation Factory Service URL.
-x, --proxyFileName <value>	Sets the proxy file to use as the client credential.
-y, --lifetime <value>	Lifetime of delegated credential in seconds. Defaults to 43200 (which is 12 hours).
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.

Name

globus-delegation-client -- C Delegation client

```
globus-delegation-client [OPTION...] {SERVICE-SPECIFIER} {{EPR-FILENAME} | {-refresh}}
```

Description

Create or refresh delegated credentials in a service container. If the `-refresh` option is specified on the command-line, then the credential associated with an existing `DelegationService` resource is updated with a new credential. Otherwise, the `SERVICE-SPECIFIER` is interpreted as a `DelegationFactoryService` and a new `DelegationService` resource is created.

Command syntax

```
globus-delegation-client [OPTION...] {SERVICE-SPECIFIER} {{EPR-FILENAME} | {-refresh}}
```

`SERVICE-SPECIFIER`: [-s URI [-k KEY VALUE] | -e FILENAME]

`EPR-FILENAME`: Name of file to store EPR of new delegated credential.

Table 3. Common options

-a --anonymous	Use anonymous authentication. Requires either -m 'conv' or transport (https) security.
-d, --debug	Enables debug mode. In debug mode, all SOAP messages will be displayed to stderr and full WSRF Fault messages will be displayed.
-e --eprFile FILENAME	Load service EPR from FILENAME. This EPR is used to contact the WSRF service.
-h --help	Displays help information about the command.
-k --key KEYNAME VALUE	Set resource key in the service EPR to be named KEYNAME with VALUE as its value. This can be combined with -s to construct an EPR without having an xml file on hand. The KEYNAME is a QName string in the format {namespaceURI}localPart . while the VALUE is a literal string to place in the element. For example, the option -k '{http://www.globus.org}MyKey' 128 would be rendered as <MyKey xmlns="http://www.globus.org">128</MyKey>
-m, --securityMech TYPE	Set authentication mechanism. TYPE is one of msg for WS-SecureMessage or conv for WS-SecureConversation.
-p, --protection LEVEL	Set message protection level. LEVEL is one of sig for digital signature or enc for encryption. The default is 'sig'.
-s --service ENDPOINT	Set ENDPOINT the service URL to use. Will be composed with the -k parameter if present to add ReferenceProperties to the ENDPOINT
-t --timeout SECONDS	Set client timeout to SECONDS.
-u --usage	Print short usage message.
-V --version	Show version information and exit.
-v --certKeyFiles CERTIFICATE-FILENAME KEY-FILENAME	Use credentials located in CERTIFICATE-FILENAME and KEY-FILENAME . The key file must be unencrypted.
-x --proxyFilename FILENAME	Use proxy credentials located in FILENAME .
-z --authorization TYPE	Set authorization mode. TYPE can be self , host , none , or a string specifying the identity of the remote party. The default is self .
--versions	Show version information for all loaded modules and exit.

Table 4. Application-specific options

-g --delegation MODE	Set the delegation mode. MODE can be 'limited' or 'full'. The default is 'limited'
-r --refresh	Refresh a credential instead of creating a new delegated credential resource.

Examples

Create a new delegated credential resource and store the EPR of the resource in `~/ .globus/delegation.epr`:

```
% globus-delegation-client -z host -s https://gridhost.virtual.org:8443/wsrf/services/Dele
```

Refresh the previously delegated credential:

```
% globus-delegation-client -z host -e ~/delegation.epr -refresh
```

Destroy the delegated credential:

```
% globus-wsrf-destroy -z host -e ~/delegation.epr
```

DRAFT

Name

`wsrf-destroy --` Destroys a resource

`wsrf-destroy`

Tool description

Destroys a resource.

Command syntax

`wsrf-destroy [options]`

DRAFT

Table 5. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: <pre>-k "{http://www.globus.org}MyKey" 123</pre>
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Example:

```
$ wsrfl-destroy -s http://localhost:8080/wsrfl/services/CounterService \ -k
  "{http://counter.com}CounterKey" 123
```

Name

`wsrf-query --` Performs query on a resource property document

`wsrf-query`

Tool description

Queries the resource property document of a resource. By default, a simple XPath query is assumed that returns the entire resource property document.

Command syntax

```
wsrf-query [options] [query expression] [dialect]
```

DRAFT

Table 6. Common options

-h, --help	Displays help information about the command.
-d, --debug	Enables debug mode. For example, full stack traces of errors will be displayed.
-e, --eprFile <file>	Specifies an <i>XML</i> file that contains the <i>WS-Addressing</i> endpoint reference.
-s, --service <url>	Specifies the service URL.
-k, --key <name value>	Specifies the resource key. The name is the QName of the resource key in the string form: {namespaceURI}localPart , while the value is the simple value of the key. For complex keys, use the --eprFile option. Example: <pre>-k "{http://www.globus.org}MyKey" 123</pre>
-f, --descriptor <file>	Specifies a client security descriptor. Overrides all other security settings.
-a, --anonymous	Enables anonymous authentication. Only supported with transport security or the GSI Secure Conversation authentication mechanism.
-g, --delegation <mode>	Enables delegation. mode can be either 'limited' or 'full' . Only supported with the GSI Secure Conversation authentication mechanism.
-l, --contextLifetime <value>	Sets the lifetime of the client security context. value is in milliseconds. Only supported with the GSI Secure Conversation authentication mechanism.
-m, --securityMech <type>	Specifies the authentication mechanism. type can be 'msg' for GSI Secure Message, or 'conv' for GSI Secure Conversation.
-c, --serverCertificate <file>	Specifies the server's <i>certificate</i> file used for encryption. Only needed for the GSI Secure Message authentication mechanism.
-p, --protection <type>	Specifies the protection level. type can be 'sig' for signature or 'enc' for encryption.
-x, --proxyFilename <value>	Sets the proxy file to use as client credential.
-z, --authorization <type>	Specifies authorization type. type can be 'self' , 'host' , 'none' , or a string specifying the expected identity of the remote party.
-t, --timeout <timeout>	Specifies client timeout (in seconds). The client will wait maximum of the timeout value for a response from the server before returning an error. By default the timeout value is 10 minutes.

Examples:

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "count(//*[local-name()='Entry'])"
```

```
$ wsrif-query -s https://127.0.0.1:8443/wsrif/services/DefaultIndexService \
  "number(//*[local-name()='GLUECE']/glue:ComputingElement/glue:State/@glue:FreeCPUs)=0"
```

```
$ wsrif-query -s http://localhost:8080/wsrif/services/ContainerRegistryService \
  "/*//*//*/*[local-name()='Address']"
```

Chapter 7. Configuring

1. Configuration overview

The security settings for Delegation Factory Service and Delegation Service can be configured by modifying the [security descriptors](#). The descriptors allow for configuring the credentials that will be used by the services and the type of authentication and message protection required, as well as the authorization mechanism.

By default, the following configuration is installed:

- Delegation Factory Service:
 - Credentials are determined by the container-level security descriptor. If there is no container-level security descriptor or if it does not specify which credentials to use, then default credentials are used.
 - Authentication and message integrity protection is enforced for the `requestSecurityToken` operation. Other operations do not require authentication. This means that you may use any of GSI *Transport*, GSI Secure Message or GSI Secure Conversation when invoking the `requestSecurityToken` operation on the Delegation Factory Service.
 - Access is authorized using the gridmap mechanism and no gridmap is configured in the service by default. If a gridmap is configured in the container-level security descriptor, it is used. To configure a *grid map file* for this service, refer to instructions in the next section.
- Delegation Service
 - Credentials are determined by the container-level security descriptor. If there is no container-level security descriptor or if it does not specify which credentials to use, then default credentials are used.
 - Authentication and message integrity protection is enforced for all operations. This means that you may use any of GSI Transport, GSI Secure Message or GSI Secure Conversation when interacting with the Delegation Service.
 - Access to resources managed by the Delegation Service is managed using the gridmap mechanism. The gridmap used is resource-specific and is populated with the subject of the client that originally created the resource. This implies that only the user who delegated can access (and refresh) the delegated credential.



Note

Changing required authentication and authorization methods will require corresponding changes to the clients that contact this service.



Important

If the service is configured to use GSI Secure Transport, then container credentials are used for the handshake, irrespective of whether service-level credentials are specified.

2. Syntax of the interface

To alter the security descriptor configuration refer to [Security Descriptors](#).

To alter the security configuration of the Delegation Factory Service, edit the file `$GLOBUS_LOCATION/etc/globus_delegation_service/factory-security-config.xml`.



Note

To either specify a gridmap file different from the container level configuration or to add one if the container security descriptor does not specify one, refer to [Section 1, “Configuring Default GridMap Files”](#) to add a gridmap to the Delegation Factory security descriptor.

To alter the security configuration of the Delegation Service, edit the file `$GLOBUS_LOCATION/etc/globus_delegation_service/service-security-config.xml`

DRAFT

Chapter 8. Environment variable interface

1. Environmental variables for Message/Transport-level Security

Refer to [Configuring](#) for environment variables. Note that the above environment variable [fixme - not clear which envvar you mean] does not supersede any settings provided in security descriptors.

DRAFT

Chapter 9. Debugging

Log output from the Delegation Service is a useful tool for debugging issues. Because the Delegation Service is built on top of Java WS Core, developer debugging is the same as described in [Debugging](#).

1. Logging in Java WS Core

The following information applies to Java WS Core and all services built on Java WS Core.

Java WS Core server side has two types of loggers. One logger is used for development logging and by default writes to standard out. The other logger includes system administration information and is [CEDPs best practices](#)¹ compliant.

On client side, only developer logging is available and is configured using `log4j.properties`.

1.1. Development Logging in Java WS Core

The following information applies to Java WS Core and those services built on it.

Logging in the Java WS Core is based on the [Jakarta Commons Logging](#)² API. Commons Logging provides a consistent interface for instrumenting source code while at the same time allowing the user to plug-in a different logging implementation. Currently we use [Log4j](#)³ as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)⁴.

1.1.1. Configuring server side developer logs

Server side logging can be configured in `$GLOBUS_LOCATION/container-log4j.properties`, when the container is stand alone container. For tomcat level logging, refer to [Logging for Tomcat](#)⁵. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

Additional logging can be enabled for a package by adding a new line to the configuration file. Example:

```
#for debug level logging from org.globus.package.FooClass
log4j.category.org.globus.package.name.FooClass=DEBUG
#for warnings from org.some.warn.package
log4j.category.org.some.warn.package=WARN
```

1.1.2. Configuring client side developer logs

Client side logging can be configured in `$GLOBUS_LOCATION/log4j.properties`. The logger `log4j.appender.A1` is used for developer logging and by default writes output to the system output. By default it is set for all warnings in the Globus Toolkit package to be displayed.

¹ <http://cedps.net/index.php/LoggingBestPractices>

² <http://jakarta.apache.org/commons/logging/>

³ <http://logging.apache.org/log4j/>

⁴ [http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure\(java.lang.String,org.apache.log4j.spi.LoggerRepository\)](http://logging.apache.org/log4j/docs/api/org/apache/log4j/PropertyConfigurator.html#doConfigure(java.lang.String,org.apache.log4j.spi.LoggerRepository))

⁵ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

1.2. Configuring system administration logs

The specific logger to edit will be `log4j.logger.sysadmin` in `$GLOBUS_LOCATION/container-log4j.properties`. There you can configure the following properties:

```
log4j.appender.infoCategory=org.apache.log4j.RollingFileAppender
log4j.appender.infoCategory.Threshold=INFO
log4j.appender.infoCategory.File=var/containerLog
log4j.appender.infoCategory.MaxFileSize=10MB
log4j.appender.infoCategory.MaxBackupIndex=2
```

Above implies the logging file is rolling with each file size limited to 10MB and the logging information is stored in `$GLOBUS_LOCATION/var/containerLog`.

1.3. Sample log file

The [sample log file](#)⁶ contains many log entries for various scenarios in the Java WS container.

2. Enabling verbose logging

As described in the above section, configuration files need to be edited to enable logging at different levels. For example, to see all debug logging statements for the Delegation Service, the following lines need to be added:

```
log4j.category.org.globus.delegation.service=DEBUG
log4j.category.org.globus.delegation.factory=DEBUG
```

3. Debugging info from delegation clients

Debugging information from delegation clients can be obtained by setting the following line in the client side logging configuration file:

```
log4j.category.org.globus.delegation.client=DEBUG
```



Note

Client-side logging configuration must be done in `$GLOBUS_LOCATION/log4j.properties`.

⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.0/common/javawscore/sample-container-log.txt>

Chapter 10. Troubleshooting




For a list of common errors in GT, see [Error Codes](#).

DRAFT

1. Error Messages

DRAFT

Table 10.1. WS A&A Delegation Service Error Messages

Error Code	Definition	Possible Solutions
<p>AuthorizationException: "test DN" is not authorized to use operation: {http://www.globus.org/08/2004/delegationService}requestSecurityToken</p>	<p>This exception can occur when a client whose DN is not in the <i>grid map file</i> configured for the delegation factory service attempts to delegate (using <code>globus-credential-delegate</code>) a credential to the factory service.</p> <p> Note</p> <p>The <i>test DN</i> specified in the error message is just a placeholder and will contain the DN of the user attempting to access the credential.</p>	<p>Ensure that the client is authorized to access delegation service. This requires the client DN to be added in the gridmap file.</p>
<p>AuthorizationException: "test DN" is not authorized to use operation: {http://www.globus.org/08/2004/delegationService}refresh</p>	<p>This exception can occur when a client attempts to refresh a credential it did not delegate (using <code>globus-credential-refresh</code>).</p> <p> Note</p> <p>The <i>test DN</i> specified in the error message is just a placeholder and will contain the DN of the user attempting to access the credential.</p>	<p>This is a delegation service policy and only client who delegates can refresh the credential.</p>
<p><i>test user DN</i> is not authorized to access this resource</p>	<p>Similar to above error but experienced by developers using the API - Only the user who created the delegated credential is allowed to access it. There are two sets of API functions for getting the credential and registering listeners: one in which the caller's DN is picked up from the current thread and the other in which a JAAS subject (containing the caller's DN) is explicitly passed as a function parameter. If the caller's DN (picked up from thread or specified explicitly) does not match the DN of the user who created the credential, this error is thrown.</p> <p> Note</p> <p>The <i>test DN</i> specified in the error message is just a placeholder and will contain the DN of the user attempting to access the credential.</p>	<p>Ensure that the DN explicitly specified or the client DN associated with the thread matches the creator's DN.</p>
<p>Unable to retrieve caller DN, cannot register</p>	<p>Developers come across this error when attempting to register a listener with a delegated credential resource without a JAAS subject. There are two ways of registering: either the JAAS subject can be explicitly passed using the API or the JAAS subject can be picked up from the current message context (the subject representing the client). If the latter mechanism for registering is used and there is no client credential associated with the thread that is calling the register function, then this exception is thrown.</p>	<p>Make sure to use the API call that explicitly passes the subject.</p>

2. CoG Configuration and troubleshooting

Also, for security-related troubleshooting, the [CoG FAQ](#)¹ might prove useful (especially sections on configuring credentials, CAs and so on).

DRAFT

¹ <http://www.globus.org/cog/distribution/1.2/FAQ.TXT>

Chapter 11. Related Documentation

- [WS-Security](#)¹
- [WS-Security: X.509 Certificate Tokens](#)²
- [WS-Trust](#)³
- [RFC 3820](#)⁴ Proxy Certificates

DRAFT

¹ <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

² <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0.pdf>

³ <http://www-106.ibm.com/developerworks/library/specification/ws-trust/>

⁴ <http://www.faqs.org/rfcs/rfc3820.html>

Glossary

C

certificate A public key plus information about the certificate owner bound together by the digital signature of a CA. In the case of a CA certificate, the certificate is self signed, i.e. it was signed using its own private key.

E

End Entity Certificate (EEC) A certificate belonging to a non-CA entity, e.g. you, me or the computer on your desk.

G

grid map file A file containing entries mapping certificate subjects to local user names. This file can also serve as a access control list for GSI enabled services and is typically found in `/etc/grid-security/grid-mapfile`. For more information see the Gridmap section [here](#).

P

private key The private part of a key pair. Depending on the type of certificate the key corresponds to it may typically be found in `$HOME/.globus/userkey.pem` (for user certificates), `/etc/grid-security/hostkey.pem` (for host certificates) or `/etc/grid-security/<service>/<service>key.pem` (for service certificates).

For more information on possible private key locations see [this](#).

proxy certificate A short lived certificate issued using a EEC. A proxy certificate typically has the same effective subject as the EEC that issued it and can thus be used in its place. GSI uses proxy certificates for single sign on and delegation of rights to other entities.

For more information about types of proxy certificates and their compatibility in different versions of GT, see <http://dev.globus.org/wiki/Security/ProxyCertTypes>.

proxy credentials The combination of a proxy certificate and its corresponding private key. GSI typically stores proxy credentials in `/tmp/x509up_u<uid>`, where `<uid>` is the user id of the proxy owner.

public key The public part of a key pair used for cryptographic operations (e.g. signing, encrypting).

S

service certificate A EEC for a specific service (e.g. FTP or LDAP). When using GSI this certificate is typically stored in `/etc/grid-security/<service>/<ser-`

`vice>cert.pem`. For more information on possible service certificate locations, see [this](#).

T

transport-level security

Uses transport-level security (TLS) mechanisms.

W

Web Services Addressing (WSA)

The WS-Addressing specification defines transport-neutral mechanisms to address web services and messages. Specifically, it defines XML elements to identify web service endpoints and to secure end-to-end endpoint identification in messages. See the [W3C WS Addressing Working Group](#)¹⁴ for details.

X

XML

Extensible Markup Language (XML) is standard, flexible, and extensible data format used for web services. See the [W3C XML site](#)²⁰ for details.

¹⁴ <http://www.w3.org/2002/ws/addr/>

²⁰ <http://www.w3.org/XML/>