

GT4 Delegation Service Admin Guide

DRAFT

GT4 Delegation Service Admin Guide

Introduction

This guide contains advanced configuration information for system administrators working with the Delegation Service. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

Important

The Delegation Service is built, installed and deployed in a default GT installation. Read [Installing GT 4.2.0](#) for more details.

DRAFT

Table of Contents

1. Building and Installing	1
1. Building and Installing Delegation Service only	1
2. Configuring	2
1. Configuration overview	2
2. Syntax of the interface	2
3. Testing	4
4. Security Considerations	5
1. Delegation Service Security Considerations	5
5. Debugging	6
1. Logging in Java WS Core	6
6. Troubleshooting	8
1. Error Messages	9
Glossary	11

DRAFT

List of Tables

6.1. WS A&A Delegation Service Error Messages 10

DRAFT

Chapter 1. Building and Installing

The delegation service is built, installed and deployed as part of GT installation. Refer to [Installing GT 4.2.0](#) for installation instructions.

1. Building and Installing Delegation Service only

This section describes installing the Delegation Service from source. The Delegation Service is a Java-only component and does not require the whole GT installation. The only dependency is on Java WS Core. The following steps describe building the Delegation Service from source and assumes Java WS Core has been installed in `GLOBUS_LOCATION`.

Important

Credentials and a gridmap file must be configured for the Delegation Service to work. Refer to [Basic Security Configuration](#) to set up credentials and gridmap files for authorization.

- Delegation Service source code can be downloaded from remote CVS and access details are provided [here](#).¹ The module name is `ws-delegation`:

```
cvs -d :pserver:anonymous@cvs.globus.org:/home/globdev/CVS/globus-packages login
cvs co -r globus_4_2_branch ws-delegation
```

- Set `GLOBUS_LOCATION` to point to your Java WS Core installation:

```
export GLOBUS_LOCATION=/path/to/core/install
```

- Run the following commands:

```
cd ws-delegation
ant deploy
```

¹ <http://www.globus.org/toolkit/docs/development/remote-cvs.html>

Chapter 2. 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 3. Testing

- Install the Delegation Service test package using GPT build.
- To run the tests, use the following commands:

```
$ cd $GLOBUS_LOCATION
$ ant -f share/globus_wsrf_test/runtests.xml runServer \
      -Dtests.jar=$GLOBUS_LOCATION/lib/globus_delegation_test.jar \
      -Djunit.jvmarg="-Dorg.globus.wsrf.container.server.id=delegationTest"
```

- The test report can be found in `$GLOBUS_LOCATION/share/globus_wsrf_test/tests/test-reports/TEST-org.globus.delegation.service.PackageTests.xml`.

Chapter 4. Security Considerations

1. Delegation Service Security Considerations

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.

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.

Chapter 5. Debugging

Because the Delegation Service is built on Java WS Core, it uses the same system admin logging, described below:

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.

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

Chapter 6. Troubleshooting

Refer to the [Globus Toolkit Administrator Guide - Security Overview](#) for details on some common security installation issues.

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

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




DRAFT

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

1. Error Messages

DRAFT

Table 6.1. 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 <u>globus-credential-delegate</u>) 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 <u>globus-credential-refresh</u>).</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>

Glossary

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

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>.

T

transport-level security Uses transport-level security (TLS) mechanisms.