

GT 4.2.0 Java WS A&A Admin Guide

DRAFT

GT 4.2.0 Java WS A&A Admin Guide

Introduction

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

Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the [Installing GT 4.2.0](#). Read through this guide before continuing!

Authentication/message protection issues. The main administration issues for authentication/message-protection deal with configuring credential-related settings. There are multiple mechanisms for doing this:

- Security Descriptors (This is the *preferred* mechanism)
 - Container Security Descriptor
 - Service Security Descriptor
- CoG properties
- Environment variables
- Relying on default behavior. The only default behaviors available concern the proxy file and trusted certificates locations.

Authorization issues. Authorization in Java WS A&A is enforced on the server and the client side. Admin configuration could include determining the container/service level authorization mechanism and setting up and managing authorization policy, e.g. entries in *grid map file* and so on. The [Chapter 2, Configuring](#) chapter describes how to configure Java WS A&A descriptors.

Table of Contents

- 1. Building and Installing 1
- 2. Configuring 2
 - 1. Configuring authorization 2
 - 2. Configuring authentication/message protection 3
- 3. Deploying 4
- 4. Testing 5
- 5. Security Considerations 8
 - 1. Security considerations for Java WS A&A 8
- 6. Debugging 9
 - 1. Logging in Java WS Core 9
- 7. Troubleshooting 11
 - 1. Credential Troubleshooting 11
 - 2. Error Messages For Java WS A&A 14
 - 3. For more troubleshooting 16
- Glossary 17

DRAFT

List of Tables

2.1. Configuring server side authentication and message/transport security	3
7.1. Credential Errors	12
7.2. Java WS A&A Errors	15

DRAFT

Chapter 1. Building and Installing

This component is built and installed as a part of Java WS Core. See "Building and Installing" in the [Java WS Core Admin Guide](#) for more information.

DRAFT

Chapter 2. Configuring

Java WS A&A is configured using security descriptors. The following describes configuration settings specific for authorization and authentication. You can read the entire Java WS A&A Security Descriptor documentation [here](#).

- [Configuring authorization](#)
- [Configuring authentication/message protection](#)

1. Configuring authorization

1.1. Configuration overview

Security descriptors are mechanisms used to configure authorization mechanism and policy. The authorization on the server side can be configured at the container, service or resource level.

On the client side, authorization can be configured using security descriptors or as a property on the stub. This configuration can be done on a per invocation granularity

1.2. Server side authorization

The server side authorization can be configured at the container, service or resource level using

- Security descriptors using files. Refer to [Section 1, “Configuring Using Files”](#)
- Security descriptors programmatically. Refer to [Section 2, “Configuring Programmatically ”](#)

To write and configure a server-side custom authorization mechanism refer to [Section 2.3, “Writing a custom server-side authorization mechanism”](#).

1.3. Client side authorization

The client side authorization can be configured for each invocation.

- Security descriptors using files. Refer to [Section 1, “Configuring Using Files”](#), specifically [Section 1.2.2, “Configuring authorization mechanism ”](#).
- Security descriptors programmatically. Refer to [Section 2, “Configuring Programmatically ”](#)
- Properties on the Stub. Refer to [Section 2.1, “Configuring client-side authorization on the stub”](#)

To write and configure custom authorization mechanism refer to [Section 2.2, “Writing custom client-side authorization scheme”](#).

If no authorization mechanism has been specified, HostOrSelf authorization is used. In this scheme host authorization is tried first, if it fails, self authorization is attempted

2. Configuring authentication/message protection

2.1. Configuration overview

Configuration of service-side security settings can be achieved by using container or service security descriptor. Some of the security configuration, like the credential to use and trusted certificates location, can also be configured using CoG properties or rely on default location. **The preferred way is to provide these settings in a security descriptor.**

The next section provides details on the relevant properties. An overview of the syntax of security descriptors can be found in [Java WS A&A Security Descriptor Framework](#). Available CoG security properties can be found in [Chapter 2, Configuring](#)

2.2. Syntax of the interface

The following properties are relevant to authentication and message/transport security:

Table 2.1. Configuring server side authentication and message/transport security

Number	Task	Descriptor Configuration	Alternate Configuration
1	Credentials	Container or service descriptor configuration	<ul style="list-style-type: none"> • X509_USER_CERT or CoG Configuration: User certificate configuration • X509_USER_KEY or CoG Configuration: User key configuration • X509_USER_PROXY or CoG Configuration: User proxy configuration <p>If no explicit configuration is found, the default proxy is read from /tmp/x509_up_<uid>.</p>
2	Trusted Certificates	Container security descriptor configuration	CoG Configuration
3	Limited proxy policy configuration	Container or service descriptor configuration	None.
4	Replay Attack Window	Container or service descriptor configuration	None.
5	Replay Attack Filter	Container or service descriptor configuration	None.
6	Replay timer interval	Container descriptor configuration	None.
7	Context timer interval	Container descriptor configuration	None.

Chapter 3. Deploying

This component is deployed as a part of [Java WS Core](#).

DRAFT

Chapter 4. Testing

To execute security tests ensure that [Ant with JUnit is configured](#).

All the security tests require a valid credential. Refer to [Security section of GT Administrator guide](#) for details on acquiring credentials.

The security tests are included in `$GLOBUS_LOCATION/lib/wsrf_test_unit.jar`. This jar contains tests for both the Java WS Core component and the WS Authentication and Authorization components contained in the Java WS Core package.

To execute the tests, pass the above jar file to the test script as described in [Section 12.3, “How do I run my JUnit tests for Java WS Core and/or my services?”](#). To ensure that only security tests are run, set `-DsecurityTestsOnly=true`.

By default the tests require that the container and the tests use the same credentials, i.e self authorization is done on secure calls.

The tests allow for another configuration in which the container can be configured with [host credentials](#) and the tests can be run with any credentials.

- Configure the container to use host credentials using the security descriptor as described in the [container descriptor](#) section.
- Edit `$GLOBUS_LOCATION/etc/globus_wsrf_test_unit/server-config.wsdd`.
 - Comment out the configured descriptor in `SecurityTestService`, `RPPParamTestService` and `AuthzCalloutTestService` that specifies self authorization.

```
<!-- Does self authz by default -->
<!-- parameter name="securityDescriptor"
      value="@config.dir@/security-config.xml" / -->
```

- Uncomment the configuration for identity authorization.

```
<!-- For use only when identity authz is used-->
<parameter name="securityDescriptor"
      value="@config.dir@/identity-security-config.xml" />
```

- In `$GLOBUS_LOCATION/etc/globus_wsrf_test_unit/identity-security-config.xml`, set the value of the property `identity` to the subject DN of the credentials used to run the tests.

```
<!-- set to expected identity -->
<param:parameter name="identity"
      value="Identity used by client" />
```

- Edit `$GLOBUS_LOCATION/etc/globus_wsrf_test_unit/test-container-security-config.xml` to add the following element, before `<reject-limited-proxy value="true"/>`

```

<credential>
  <cert-key-files>
    <key-file value="path to container key"/>
    <cert-file value="path to container cert"/>
  </cert-key-files>
</credential>

```

- Start a secure and insecure standalone container.

```

$ cd $GLOBUS_LOCATION
$ bin/globus-start-container -nosec

```

On another window,

```

$ cd $GLOBUS_LOCATION
$ bin/globus-start-container

```

- To run tests against external containers, secure and insecure, on localhost ports 8180 and 8181 respectively, the command would be:

```

ant -f share/globus_wsrf_test/runtests.xml runServer
  -Dtests.jar=$GLOBUS_LOCATION/lib/wsrf_test_unit.jar
  -DsecurityTestsOnly=true
  -Djunit.jvmarg=-Dsecurity.test.client.authz=host
  -Dsecurity.test.authz.identity="container/suject/DN"
  -Dsecurity.test.enc.cred="/server/public/certificate"
  -Dsecurity.test.server.cert="/server/public/certificate/file"
  -Dsecurity.test.server.key="/server/key/file"
  -Dtest.server.url=http://127.0.0.1:8181/wsrf/services/
  -Dsecure.test.server.url=https://127.0.0.1:8180/wsrf/services/

```

- For example, if the container security descriptor has credentials configured as */etc/grid-security/containercert.pem* and */etc/grid-security/containerkey.pem* and the DN of the credential is */DC=org/OU=Services/CN=-testDN/CN=some.host.edu*, the command to run would be:

```

ant -f share/globus_wsrf_test/runtests.xml runServer
  -Dtests.jar=$GLOBUS_LOCATION/lib/wsrf_test_unit.jar
  -DsecurityTestsOnly=true
  -Djunit.jvmarg=-Dsecurity.test.client.authz=host
  -Dtest.server.url=http://127.0.0.1:8181/wsrf/services/
  -Dsecure.test.server.url=https://127.0.0.1:8180/wsrf/services/
  -Dsecurity.test.authz.identity="/DC=org/OU=Services/CN=-testDN/CN=some.host.edu"
  -Dsecurity.test.server.cert="/etc/grid-security/containercert.pem"

```

```
-Dsecurity.test.server.key="/etc/grid-security/containerkey.pem"  
-Dsecurity.test.enc.cred="/etc/grid-security/containercert.pem"
```

DRAFT

Chapter 5. Security Considerations

1. Security considerations for Java WS A&A

1.1. Security considerations for authorization

1.1.1. Client side authorization

Client authorization of the server is done after the completion of the operation when GSI Secure Message authentication is used. If you require client authorization to be done prior, use GSI Secure Conversation or GSI *Transport security*.

1.1.2. Host authorization

During host authorization, the toolkit treats DNS "hostname-*.edu" as equivalent to "hostname.edu". This means that if a service was setup to do host authorization and hence accept the certificate "hostname.edu", it would also accept certificates with DNS "hostname-*.edu".

The feature is in place to allow a multi-homed host following a "hostname-interface" naming convention, to have a single host certificate. For example, host "grid.test.edu" would also accept likes of "grid-1.test.edu" or "grid-foo.test.edu".

Note

The wildcard character "*" matches only name of the host and not domain components. This means that "hostname.edu" will not match "hostname-foo.sub.edu", but will match "host-foo.edu".

Note

If a host was set up to accept "hostname-1.edu", it will not accept any of "hostname-*.edu".

A [bug](#)¹ has been opened to see if this feature needs to be modified.

1.2. Security considerations for Message/Transport-level Security

1.2.1. File permissions

The Java security code currently does not enforce secure permissions and, implicitly, file ownership requirements on any of the security related files, e.g. configuration and credential files. It is thus important that administrators ensure that the relevant files have correct permissions and ownership. Permissions should generally be as restrictive as possible, i.e. *private keys* should be readable only by the file owner and other files should be writable by owner only, and the files should generally be owned by the globus user (the requirements that the C code enforces are documented in [Configuring GSI](#)).

Also refer to [Section 5, "Known Problems"](#) for details on any other open issues.

¹ http://bugzilla.globus.org/bugzilla/show_bug.cgi?id=2969

Chapter 6. Debugging

Because Java WS A&A is built on Java WS Core, it uses the same sys 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 7. Troubleshooting

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

1. Credential Troubleshooting

1.1. Credential Errors

The following are some common problems that may cause clients or servers to report that credentials are invalid:

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

DRAFT

Table 7.1. Credential Errors

Error Code	Definition	Possible Solutions
Your proxy credential may have expired	Your proxy credential may have expired.	Use <code>grid-proxy-info</code> to check whether the proxy credential has actually expired. If it has, generate a new proxy with <code>grid-proxy-init</code> .
The system clock on either the local or remote system is wrong.	This may cause the server or client to conclude that a credential has expired.	Check the system clocks on the local and remote system.
Your end-user certificate may have expired	Your end-user certificate may have expired	Use <code>grid-cert-info</code> to check your certificate's expiration date. If it has expired, follow your CA's procedures to get a new one.
The permissions may be wrong on your proxy file	If the permissions on your proxy file are too lax (for example, if others can read your proxy file), Globus Toolkit clients will not use that file to authenticate.	You can "fix" this problem by changing the permissions on the file or by destroying it (with <code>grid-proxy-destroy</code>) and creating a new one (with <code>grid-proxy-init</code>). Important: However, it is still possible that someone else has made a copy of that file during the time that the permissions were wrong. In that case, they will be able to impersonate you until the proxy file expires or your permissions or end-user certificate are revoked, whichever happens first.
The permissions may be wrong on your private key file	If the permissions on your end user certificate private key file are too lax (for example, if others can read the file), <code>grid-proxy-init</code> will refuse to create a proxy certificate.	You can "fix" this by changing the permissions on the private key file. Important: However, you will still have a much more serious problem: it is possible that someone has made a copy of your private key file. Although this file is encrypted, it is possible that someone will be able to decrypt the private key, at which point they will be able to impersonate you as long as your end user certificate is valid. You should contact your CA to have your end-user certificate revoked and get a new one.
The remote system may not trust your CA	The remote system may not trust your CA	Verify that the remote system is configured to trust the CA that issued your end-entity certificate. See Installing GT 4.2.0 for details.
You may not trust the remote system's CA	You may not trust the remote system's CA	Verify that your system is configured to trust the remote CA (or that your environment is set up to trust the remote CA). See Installing GT 4.2.0 for details.
There may be something wrong with the remote service's credentials	There may be something wrong with the remote service's credentials	It is sometimes difficult to distinguish between errors reported by the remote service regarding your credentials and errors reported by the client interface regarding the remote service's credentials. If you cannot find anything wrong with your credentials, check for the same conditions on the remote system (or ask a remote administrator to do so).

1.2. Some tools to validate certificate setup

1.2.1. Check that the user certificate is valid

```
openssl verify -CApath /etc/grid-security/certificates
-purpose sslclient ~/.globus/usercert.pem
```

1.2.2. Connect to the server using s_client

```
openssl s_client -ssl3 -cert ~/.globus/usercert.pem -key
~/.globus/userkey.pem -CApath /etc/grid-security/certificates
-connect <host:port>
```

Here `<host:port>` denotes the server and port you connect to.

If it prints an error and puts you back at the command prompt, then it typically means that the *server* has closed the connection, i.e. that the server was not happy with the client's certificate and verification. Check the SSL log on the server.

If the command "hangs" then it has actually opened a telnet style (but secure) socket, and you can "talk" to the server.

You should be able to scroll up and see the subject names of the server's verification chain:

```
depth=2 /DC=net/DC=ES/O=ESnet/OU=Certificate Authorities/CN=ESnet Root CA 1
verify return:1
depth=1 /DC=org/DC=DOEGrids/OU=Certificate Authorities/CN=DOEGrids CA 1
verify return:1
depth=0 /DC=org/DC=doegrids/OU=Services/CN=wiggum.mcs.anl.gov
verify return:1
```

In this case, there were no errors. Errors would give you an extra line next to the subject name of the certificate that caused the error.

1.2.3. Check that the server certificate is valid

Requires root login on server:

```
openssl verify -CApath /etc/grid-security/certificates -purpose sslserver
/etc/grid-security/hostcert.pem
```

2. Error Messages For Java WS A&A

DRAFT

Table 7.2. Java WS A&A Errors

Error Code	Definition	Possible Solutions
[JWSSEC-248] Secure container requires valid credentials	This error occurs when <code>globus-start-container</code> is run without any valid credentials. Either a proxy certificate or service/host certificate needs to be configured for the container to start up.	<ol style="list-style-type: none"> <li data-bbox="805 300 1334 604">1. If you are not looking to start up a container that uses GSI Secure Transport, which is used by the container by default, use <code>globus-start-container -nosec</code>. You will be able to use insecure clients and services. However, this also implies that if you have not configured individual services with credentials, you will not be able to securely access the service. <li data-bbox="805 615 1334 804">2. If you are running a personal container, generate a proxy certificate with <code>grid-proxy-init</code>. If the proxy certificate is not in the default location, configure the container security descriptor as described in Configuring Container Security Descriptor. <li data-bbox="805 825 1334 930">3. If you want to use host certificates, configure the container security descriptor as described Configuring Credentials.
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-250] Failed to load certificate/key file]	This error occurs if the file path to the container certificate and key configured are invalid.	<ol style="list-style-type: none"> <li data-bbox="805 963 1334 1173">1. The path to the container certificate and key are configured in <code>\$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml</code>. This file is loaded as described [here - fixme link]. Ensure that the path is correct.
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-249] Failed to load proxy file]	This error occurs if container proxy file configured is invalid.	<ol style="list-style-type: none"> <li data-bbox="805 1194 1334 1404">1. The path to the container proxy certificates are configured in <code>\$GLOBUS_LOCATION/etc/globus_wsrf_core/global_security_descriptor.xml</code>. This file is loaded as described [here - fixme link]. Ensure that the path is correct.
Failed to start container: Container failed to initialize [Caused by: [JWSSEC-245] Error parsing file: "etc/globus_wsrf_core/global_security_descriptor.xml" [Caused by: ...]	This error occurs if the container security descriptor configured is invalid.	<ol style="list-style-type: none"> <li data-bbox="805 1425 1334 1530">1. The container security descriptor should conform to the Container Security Descriptor Schema.¹ <li data-bbox="805 1541 1334 1614">2. Refer to the "Caused by: " section for details on the specific element that is not correct.

¹ http://www.globus.org/toolkit/docs/4.2.0/security/container_security_descriptor.xsd

Error Code	Definition	Possible Solutions
[JGLOBUS-77] Unknown CA	This error occurs if the CA certificate for the credentials being used is not installed correctly.	<ol style="list-style-type: none">1. If this issue occurs on the server side, the container is not configured with CA certificates. The container looks for trusted certificates in the default location as described Java CoG Toolkit FAQ²2. On the server side, the trusted certificates can be configured as described in Trusted Certificates3. On the client side, trusted certificates can be configured as described in Configuring Trusted Credentials

3. For more troubleshooting

- [Troubleshooting Java WS Core](#)
- [Globus Toolkit Administrator Guide - Security Section](#)

² <http://www.globus.org/cog/distribution/1.2/FAQ.TXT>

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](#).

H

host credentials The combination of a host certificate and its corresponding private key.

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](#).

T

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