

GT 4.2.1 WS RLS: System Administrator's Guide

GT 4.2.1 WS RLS: System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with WS RLS. It provides references to information on procedures typically performed by system administrators, including installation, 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 4.2.1](#). Read through this guide before continuing!

Table of Contents

1. Building and Installing	1
1. Installing WS RLS using Globus Toolkit Binary Bundle	1
2. Installing WS RLS using Globus Toolkit Source Bundle	1
3. Installing WS RLS using Globus CVS	2
2. Configuring WS RLS	3
1. Configuration overview	3
2. WS RLS Web Service Deployment Descriptor (WSDD) Configuration	3
3. WS RLS Java Naming & Directory Interface (JNDI) Configuration	4
3. Testing	5
4. Security Considerations	6
1. WS Replica Location Service (WS RLS) Security Considerations	6
5. Debugging	7
1. Logging in Java WS Core	7
2. Increasing log verbosity	8
6. Troubleshooting	9
1. Errors	9
7. Usage statistics collection by the Globus Alliance	10
1. WS RLS-specific usage statistics	10
Index	11

List of Tables

6.1. WS Replica Location Service (WS RLS) Errors	9
--	---

Chapter 1. Building and Installing

WS RLS is distributed with the Globus Toolkit 4.2.1 and is available in both the binary bundles and the source bundle. For most purposes, the binary bundle provides the simplest means of installing the WS RLS and its requirements. There are three typical ways to access and install the WS RLS:

- [Section 1, “Installing WS RLS using Globus Toolkit Binary Bundle”](#)
- [Section 2, “Installing WS RLS using Globus Toolkit Source Bundle”](#)
- [Section 3, “Installing WS RLS using Globus CVS”](#)

Important

Before proceeding with installation of WS RLS, please familiarize yourself with the installation procedures for RLS (see [Replica Location Service \(RLS\)](#)).

1. Installing WS RLS using Globus Toolkit Binary Bundle

Follow instructions provided by [Installing GT 4.2.1](#) and [Installing GT](#) in order to install the WS RLS from one of the available binary bundles of the Globus Toolkit.

Once you have unpackaged the binary bundle, the following commands may be used to install WS RLS:

```
% ./configure --prefix=$GLOBUS_LOCATION --enable-rls --enable-wsrls
% make
% make install
```

2. Installing WS RLS using Globus Toolkit Source Bundle

WS RLS is not installed as part of the default source install. In order to install WS RLS, you will need to ensure that RLS is also installed as part of the build by reviewing its installation procedures. Along with RLS configure settings (e.g., ODBC related settings) the `--enable-wsrls` flag is required as part of the configuration step.

The following commands will install the WS RLS (optionally you may use the "gt4-wsrls" make target to build only WS RLS and its requirements:

```
% ./configure --prefix=/path/to/install --enable-rls --enable-wsrls
% make [rls] gt4-wsrls]
% make install
```

3. Installing WS RLS using Globus CVS

Additionally, you may access the WS RLS from the Globus Toolkit CVS repository. You may deploy the 4.1.x WS RLS on top of an existing GT 4.1.x installation using trunk.

Important

Please review [Accessing the Globus Alliance's Code via CVS](#)¹ for the authoritative information on building and installing the Globus Toolkit from source in our CVS.

The following instructions may be used to build and install the WS RLS:

```
setenv CVSROOT :pserver:anonymous@cvs.globus.org:/home/globdev/CVS/globus-packages
cvs co packaging/
cd packaging
./make-packages.pl --bundles=gt4-wsrls --deps [--verbose]
setenv GPT_LOCATION /path/to/packaging/gpt-3.2globus2
$GPT_LOCATION/sbin/gpt-build [-verbose] ./bundle-output/custom-deps-1.0-src_bundle.tar.gz
```

Note

Changes to the toolkit (WS Core and WSAA) made between the 4.0.x release and the latest code base, make the GT 4.0.x build environment unsuitable for GT 4.2.1 components. As such the WS RLS cannot be deployed on an existing GT 4.0.x installation. The good news, if you will, is that the WS RLS can use your existing GT 4.0.x RLS without modification. If you have an existing RLS installation, and you want to try out the WS RLS interface layer, you need not reinstall RLS. Of course, you may want to reinstall RLS exclusively for testing with the WS RLS interface, but it is up to your own choosing.

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

Chapter 2. Configuring WS RLS

1. Configuration overview

The WS RLS requires certain WSDD and JNDI settings to be properly configured. The installed JNDI configuration file may be found at `$GLOBUS_LOCATION/etc/globus_wsrf_replicolocation_resource/jndi-config.xml`. To view the default JNDI file (current snapshot) from the Globus CVS repository [click here](#)¹. The installed WSDD configuration file may be found at `$GLOBUS_LOCATION/etc/globus_wsrf_replicolocation_service/server-config.wsdd`. To view the default WSDD file (current snapshot) from the Globus CVS repository [click here](#)². In most cases, you will not need to alter either of these files.

This information is in addition to the basic configuration instructions in the [Installing GT 4.2.1](#).

2. WS RLS Web Service Deployment Descriptor (WSDD) Configuration

The WS RLS's WSDD determines which services will be activated by the GT Java WS Core container. By default, the WS RLS defines two services.

- `ReplicaLocationCatalogService`. The `ReplicaLocationCatalogService` service provides the WS interface to the RLS Local Replica Catalog capabilities. If you intend to activate this service, you need not alter the WSDD. If you do not wish to activate this service, simply remove it from the WSDD and restart the container.

```
...
    <service name="ReplicaLocationCatalogService" provider="Handler"
        use="literal" style="document">
...
</service>
...
```

- `ReplicaLocationIndexService`. The `ReplicaLocationIndexService` service provides the WS interface to the RLS Replica Location Index capabilities. If you intend to activate this service, you need not alter the WSDD. If you do not wish to activate this service, simply remove it from the WSDD and restart the container.

```
...
    <service name="ReplicaLocationIndexService" provider="Handler"
        use="literal" style="document">
...
</service>
...
```

¹ <http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/resource/java/source/deploy-jndi-config.xml>

² <http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/service/java/source/deploy-server.wsdd>

3. WS RLS Java Naming & Directory Interface (JNDI) Configuration

The WS RLS's JNDI configuration determines the resource class to be used for each service defined by the WS RLS WSDD. It also determines other critical settings used by these resources. In this file, you will find settings for the two default services defined in the WSDD.

Each service has a JNDI Resource of name `RLSConnectionSource`. This JNDI Resource is used by the WS RLS service to open a connection to the backend RLS service. In many cases, you may find that the default settings require no modification for your setup. The `url` parameter sets the endpoint for the backend RLS. The `certfile` and `keyfile` parameters set the certificate and key files to be used by the WS RLS to communicate securely with the backend RLS.

You will notice that the default settings use the container's suggested certificate and key file paths. If you have chosen different paths in which to keep these files, please adjust the settings accordingly. At present, the WS RLS uses a single identity (defined by the certificate and key) to authenticate itself with the RLS in order to perform any user requested operations. When the WS RLS interface is finalized, it will most likely use a different approach to authentication between WS RLS and the back end RLS. At this time, we expect that the interface will change such that the caller's credentials will be used to make calls between the WS RLS and the back end RLS on behalf of the originating caller.

```
...
  <resource
    name="RLSConnectionSource"
    type="org.globus.replica.rls.connection.impl.PooledRLSConnectionSource">
  <resourceParams>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrfl.jndi.BeanFactory</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>rls://localhost</value>
    </parameter>
    <parameter>
      <name>certfile</name>
      <value>/etc/grid-security/containercert.pem</value>
    </parameter>
    <parameter>
      <name>keyfile</name>
      <value>/etc/grid-security/containerkey.pem</value>
    </parameter>
  </resourceParams>
</resource>
...
```

Chapter 3. Testing

At this time, the WS RLS does not yet provide an automated test suite.

Chapter 4. Security Considerations

1. WS Replica Location Service (WS RLS) Security Considerations

Security recommendations include:

- The following paragraph describes our INITIAL IMPLEMENTATION in this first look of the WS RLS interface -- THIS IS NOT INTENDED TO BE A FINAL SOLUTION.

Users of the WS RLS authenticate themselves to the WS RLS in the usual manner based on their credential. The WS RLS uses the WSAA and Authorization Framework to make authorization decisions. Then, the WS RLS uses a single certificate and key file to identify itself to the RLS irregardless of the user accessing the WS RLS. Thus users that can access the WS RLS are given the fixed WS RLS identity to access the RLS. Users accustomed to using RLS may not feel comfortable with this approach -- if it may be an issue for your environment, we suggest that you not use it with your production RLS. If you think of WS RLS as the gating interface to the RLS, as you should, then you should apply the appropriate authorization restrictions at the WS RLS level, which can be done using WSAA. This will in effect achieve a level of authorization similar to that of the RLS.

Chapter 5. Debugging

The following information is from Java WS Core, on which WS RLS is based.

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. Increasing log verbosity

Generating verbose log output is a critical aid in troubleshooting of the WS RLS and is useful when communicating problems to Globus support lists. To increase logging detail, add the following lines to the `$GLOBUS_LOCATION/container-log4j.properties` file.

```
...
log4j.category.org.globus.replica.location=DEBUG
log4j.category.org.globus.replica.rls=DEBUG
...
```

⁶ <http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/sample-container-log.txt>

Chapter 6. Troubleshooting

For sys admin debugging information, see [Chapter 5, Debugging](#).

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

1. Errors

Table 6.1. WS Replica Location Service (WS RLS) Errors

Error Code	Definition	Possible Solutions
Error: java.lang.Null- PointerException	When invoking the WS RLS command-line clients, a system-level exception like the one above may be encountered.	The admin should check the container logs for the exact error.
Error: A server error occurred while processing the request	When invoking the WS RLS command-line clients, a server error like the one above may be encountered.	The admin should check the container logs for the exact error.
java.lang.UnsatisfiedLinkError	This exception when using the WS RLS may indicate that the native RLS libraries that WS RLS depends on could not be located.	To correct this problem, ensure that the <code>\$GLOBUS_LOCATION/lib</code> directory is in the library search path (on some systems this is the <code>LD_LIBRARY_PATH</code> variable).
Unable to connect to localhost:39281	The WS RLS is an interface layer that depends on the RLS for the replica location functionality. You must install and run RLS and configure WS RLS to use the RLS via its JNDI configuration.	Check that RLS is installed, running, and check that the WS RLS JNDI configuration uses the correct host-name and port to connect to the RLS.
org.globus.com-mon.ChainedIOException: Failed to initialize security context	If this exception occurs while using WS RLS, it may indicate that the user's proxy is invalid.	To correct the error, the user must properly initialize the user proxy. See grid-proxy-init for more information on proxy initialization.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	If this exception occurs when using the WS RLS, it may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered. While all the XML types should get registered upon deployment without intervention by the user, sometimes they do not.	To remedy the situation add a <code>typeMapping</code> to the <code>server-config.wsdd</code> file under <code>globus_wsrf_replicalocation_service</code> . Use the format shown here .

Chapter 7. Usage statistics collection by the Globus Alliance

1. WS RLS-specific usage statistics

The WS RLS does not collect usage statistics in addition to what the RLS collects. Please consult the RLS documentation to familiarize yourself with usage statistics collected by it.

Also, please see our [policy statement](#)¹ on the collection of usage statistics.

¹ [../../../../Usage_Stats.html](#)

Index

C

- configuring ReplicaLocationCatalogService, 3
- configuring ReplicaLocationIndexService, 3
- configuring WS RLS, 3

D

- debugging
 - logging, 7

E

- errors, 9

I

- installing
 - from binary, 1
 - from CVS, 2
 - from source, 1

L

- logging, 7
 - CEDPS-compliant, 7
 - debugging, 7

T

- troubleshooting
 - errors, 9

U

- usage statistics, 10

GT 4.2.1 WS RLS : User's Guide

GT 4.2.1 WS RLS : User's Guide

Introduction

The WS Replica Location Service (WS RLS) provides a Web services interface for the existing Replica Location Service (RLS). The features of the RLS have been preserved and interfaces should look familiar to users accustomed to working with the RLS. In most cases the interfaces are semantically similar (or identical) to the RLS operations, but effort has been made to adapt the interfaces to conform to expected style and conventions more typical of Web services. The interfaces consist of the WS-ReplicaLocationCatalog and the WS-ReplicaLocationIndex, representing the RLS Local Replica Catalog (LRC) and RLS Replica Location Index (RLI).



Note

Please note this is relatively new code and may not necessarily be as robust as the rest of RLS.

Usage of the WS-RLS is related to but not exactly similar to the RLS. Users of the RLS will find that RLS operations have been organized into a set of command-line tools. The command-line tools provide capabilities for updating replica catalogs. For instance, a tool (globus-replicolocation-createmappings) supports the creating of new logical name to target name mappings. Such tools exist for creating mappings, adding mappings, deleting mappings, defining attributes, undefining attributes, and adding attributes, modifying attributes, and removing attributes. For query operations, both against the catalog as well as the index service, the WS-RLS supports the wsrf-query command with a custom query dialect (see usage for examples of the query dialect).

Table of Contents

1. Mapping Replica Locations (WS)	1
1. Create mappings	1
2. Add mappings	1
3. Define attribute definitions	1
4. Add attributes	1
5. Query mappings	2
6. Query attributes	2
I. WS RLS Commands	?
globus-repicalocation-createmappings	4
globus-repicalocation-addmappings	5
globus-repicalocation-deletemappings	6
globus-repicalocation-defineattributes	7
globus-repicalocation-undefineattributes	8
globus-repicalocation-addattributes	9
globus-repicalocation-modifyattributes	10
globus-repicalocation-removeattributes	11
2. Troubleshooting	12
1. Errors	12
Index	13

List of Tables

1. globus-repicalocation-createmappings Options	4
2. globus-repicalocation-addmappings Options	5
3. globus-repicalocation-deletemappings Options	6
4. globus-repicalocation-defineattributes Options	7
5. globus-repicalocation-undefineattributes Options	8
6. globus-repicalocation-addattributes Options	9
7. globus-repicalocation-modifyattributes Options	10
8. globus-repicalocation-removeattributes Options	11
2.1. WS Replica Location Service (WS RLS) Errors	12

Chapter 1. Mapping Replica Locations (WS)

This section describes a few key usage scenarios and provides examples of using the WS RLS command-line tools.

1. Create mappings

Use the `globus-replicolocation-createmappings(1)` tool to create mappings.

```
% $GLOBUS_LOCATION/bin/globus-replicolocation-createmappings \  
-s https://localhost:8443/wsrp/services/ReplicaLocationCatalogService \  
mydata1 gsiftp://path/a/to/mydata1
```

No output is expected from this command when successful.

2. Add mappings

Use the `globus-replicolocation-addmappings(1)` tool to add mappings.

```
% $GLOBUS_LOCATION/bin/globus-replicolocation-addmappings \  
-s https://localhost:8443/wsrp/services/ReplicaLocationCatalogService \  
mydata1 gsiftp://path/b/to/mydata1
```

No output is expected from this command when successful.

3. Define attribute definitions

Use the `globus-replicolocation-defineattributes(1)` tool to define attribute definitions.

```
% $GLOBUS_LOCATION/bin/globus-replicolocation-defineattributes \  
-s https://localhost:8443/wsrp/services/ReplicaLocationCatalogService \  
myattr1 logical string
```

No output is expected from this command when successful.

4. Add attributes

Use the `globus-replicolocation-addattributes(1)` tool to add attributes.

```
% $GLOBUS_LOCATION/bin/globus-replicolocation-addattributes \  
-s https://localhost:8443/wsrp/services/ReplicaLocationCatalogService \  
myattr1 logical string
```

```
mydata1 myattr1 logical string attribute-value-goes-here
```

No output is expect from this command when successful.

5. Query mappings

Use the [wsrf-query](#) tool to query mappings.

```
% $GLOBUS_LOCATION/bin/wsrf-query \  
-s https://localhost:8443/wsrf/services/ReplicaLocationCatalogService \  
"query-target: mydata1" \  
"http://globus.org/replica/location/06/01/QueryDialect" \  
<ns1:MappingStatusType ns1:logical="mydata1" \  
ns1:target="gsiftp://path/a/to/mydata1" \  
xmlns:ns1="http://www.globus.org/namespaces/2005/08/replica/location"/> \  
<ns1:MappingStatusType ns1:logical="mydata1" \  
ns1:target="gsiftp://path/b/to/mydata1" \  
xmlns:ns1="http://www.globus.org/namespaces/2005/08/replica/location"/>
```

6. Query attributes

Use the [wsrf-query](#) tool to query attributes.

```
% $GLOBUS_LOCATION/bin/wsrf-query \  
-s https://localhost:8443/wsrf/services/ReplicaLocationCatalogService \  
"query-logical-attributes: mydata1" \  
"http://globus.org/replica/location/06/01/QueryDialect" \  
<ns1:AttributeStatusType ns1:key="mydata1" ns1:name="myattr1" \  
ns1:objtype="logical" ns1:status="attributeExists" ns1:valtype="string" \  
xmlns:ns1="http://www.globus.org/namespaces/2005/08/replica/location"> \  
<_value xmlns="">attribute-value-goes-here</_value> \  
</ns1:AttributeStatusType>
```

WS RLS Commands

The WS RLS provides a set of command-line tools to create, add, remove mappings between logical names and target names, define and undefine attribute definitions, and create, modify, and delete attributes. These command line tools are available on Unix and Windows platforms and will work in the same way (of course within the platform rules - the path syntax, variable definitions, etc.).

The WS RLS command-line tools make use of the Common Java Client Options. These options are referred to below as [options].

Name

`globus-replication-createmappings` -- This tool is used to create mappings between logical names and target names. The *create* semantic implies that the logical name does not exist at the time of invocation.

`globus-replication-createmappings`

Tool description

Use this tool to create mappings between logical names and target names in the replica location catalog. The mapping must not exist. In addition, the logical name must not exist.

Command syntax

```
globus-replication-createmappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 1. `globus-replication-createmappings` Options

<code>{ logical-name target-name }+</code>	A listing of logical name to target name mappings.
<code>input-file</code>	A file containing logical name to target name mappings.
<code>-</code>	Standard input stream containing logical name to target name mappings.

Name

`globus-repicalocation-addmappings` -- This tool is used to add mappings between logical names and target names. The *add* semantic implies that the logical name does exist at the time of invocation.

`globus-repicalocation-addmappings`

Tool description

Use this tool to add mappings between logical names and target names in the replica location catalog. The mapping must not exist. In addition, the logical name must exist.

Command syntax

```
globus-repicalocation-addmappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 2. `globus-repicalocation-addmappings` Options

<code>{ logical-name target-name }+</code>	A listing of logical name to target name mappings.
<code>input-file</code>	A file containing logical name to target name mappings.
<code>-</code>	Standard input stream containing logical name to target name mappings.

Name

globus-repicalocation-deletemappings -- This tool is used to delete mappings between logical names and target names.

globus-repicalocation-deletemappings

Tool description

Use this tool to delete mappings between logical names and target names in the replica location catalog. The mapping must exist.

Command syntax

```
globus-repicalocation-deletemappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 3. globus-repicalocation-deletemappings Options

{ logical-name target-name }+	A listing of logical name to target name mappings.
input-file	A file containing logical name to target name mappings.
-	Standard input stream containing logical name to target name mappings.

Name

globus-repicalocation-defineattributes -- This tool is used to define attributes.

globus-repicalocation-defineattributes

Tool description

Use this tool to define attributes. Attribute definitions must be given a name unique within the local instance of the replica location catalog. Attribute definitions must be given a value type of dateTime, decimal, integer, or string. And attribute definitions must be associated with an object type of logical or target.

Command syntax

```
globus-repicalocation-defineattributes [options] \  
{ { name object-type value-type }+ | input-file | - }
```

Table 4. globus-repicalocation-defineattributes Options

{ name object-type value-type }+	A listing of attribute name, associated object-type, and value-type.
input-file	A file containing the listing of attribute name, associated object-type, and value-type.
-	Standard input stream containing the listing of attribute name, associated object-type, and value-type.

Name

globus-repicalocation-undefineattributes -- This tool is used to undefine attributes.

globus-repicalocation-undefineattributes

Tool description

Use this tool to undefine attributes. Attribute definitions must be identified by the definition's name and associated object-type. The operation will clear attribute values for existing attributes with the definition's name.

Command syntax

```
globus-repicalocation-undefineattributes [options] \  
{ { name object-type }+ | input-file | - }
```

Table 5. globus-repicalocation-undefineattributes Options

{ name object-type }+	A listing of attribute name and associated object-type.
input-file	A file containing the listing of attribute name and associated object-type.
-	Standard input stream containing the listing of attribute name and associated object-type.

Name

globus-repicalocation-addattributes -- This tool is used to add attributes.

globus-repicalocation-addattributes

Tool description

Use this tool to add attributes associated with logical names or target names. A corresponding attribute definition must exist. The logical name or target name with which to associate the attribute must exist. There must not be an existing attribute of the same type for a given logical name or target name. When adding attributes, the following parameters are required. The logical name or target name, referred to as the key. The name of the attribute as defined by an existing attribute definition. An object-type of logical or target. A value-type corresponding to dateTime, decimal, integer, or string. And finally a value compatible with the value-type.

Command syntax

```
globus-repicalocation-addattributes [options] \  
{ { key name object-type value-type value }+ | input-file | - }
```

Table 6. globus-repicalocation-addattributes Options

{ key name object-type value-type value }+	A listing of key, attribute name, associated object-type, value-type, and value.
input-file	A file containing the listing of key, attribute name, associated object-type, value-type, and value.
-	Standard input stream containing the listing of key, attribute name, associated object-type, value-type, and value.

Name

globus-replication-modifyattributes -- This tool is used to modify attributes.

globus-replication-modifyattributes

Tool description

Use this tool to modify attributes associated with logical names or target names. Mutability of attributes is limited only to the attribute's value. The corresponding attribute must exist.

Command syntax

```
globus-replication-modifyattributes [options] \  
{ { key name object-type value-type value }+ | input-file | - }
```

Table 7. globus-replication-modifyattributes Options

{ key name object-type value-type value }+	A listing of key, attribute name, associated object-type, value-type, and value.
input-file	A file containing the listing of key, attribute name, associated object-type, value-type, and value.
-	Standard input stream containing the listing of key, attribute name, associated object-type, value-type, and value.

Name

globus-repicalocation-removeattributes -- This tool is used to remove existing attributes.

globus-repicalocation-removeattributes

Tool description

Use this tool to remove existing attributes associated with logical names or target names. The corresponding attribute must exist.

Command syntax

```
globus-repicalocation-removeattributes [options] \  
{ { key name object-type }+ | input-file | - }
```

Table 8. globus-repicalocation-removeattributes Options

{ key name object-type }+	A listing of key, attribute name, and associated object-type.
input-file	A file containing the listing of key, attribute name, and associated object-type.
-	Standard input stream containing the listing of key, attribute name, and associated object-type.

Chapter 2. Troubleshooting

For sys admin debugging information, see [Chapter 5, Debugging](#).

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

1. Errors

Table 2.1. WS Replica Location Service (WS RLS) Errors

Error Code	Definition	Possible Solutions
Error: java.lang.Null- PointerException	When invoking the WS RLS command-line clients, a system-level exception like the one above may be encountered.	The admin should check the container logs for the exact error.
Error: A server error occurred while processing the request	When invoking the WS RLS command-line clients, a server error like the one above may be encountered.	The admin should check the container logs for the exact error.
java.lang.UnsatisfiedLinkError	This exception when using the WS RLS may indicate that the native RLS libraries that WS RLS depends on could not be located.	To correct this problem, ensure that the \$GLOBUS_LOCATION/lib directory is in the library search path (on some systems this is the LD_LIBRARY_PATH variable).
Unable to connect to localhost:39281	The WS RLS is an interface layer that depends on the RLS for the replica location functionality. You must install and run RLS and configure WS RLS to use the RLS via its JNDI configuration.	Check that RLS is installed, running, and check that the WS RLS JNDI configuration uses the correct host-name and port to connect to the RLS.
org.globus.com-mon.ChainedIOException: Failed to initialize security context	If this exception occurs while using WS RLS, it may indicate that the user's proxy is invalid.	To correct the error, the user must properly initialize the user proxy. See grid-proxy-init for more information on proxy initialization.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	If this exception occurs when using the WS RLS, it may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered. While all the XML types should get registered upon deployment without intervention by the user, sometimes they do not.	To remedy the situation add a typeMapping to the server-config.wsdd file under globus_wsrf_replicalocation_service. Use the format shown here .

Index

A

- adding attributes, 9
- adding mappings, 5
- attributes
 - adding, 1
 - defining, 1
 - definitions, 1
 - querying, 2

C

- creating mappings, 4

D

- defining attributes, 7
- deleting mappings, 6

E

- errors, 12

M

- mapping
 - querying, 2
- mappings
 - adding, 1
 - creating, 1
 - querying, 2
- modifying attributes, 10

R

- removing attributes, 11

T

- troubleshooting
 - errors, 12

U

- undefining attributes, 8

GT 4.2.1 WS RLS : Developer's Guide

GT 4.2.1 WS RLS : Developer's Guide

Introduction

This guide contains information of interest to developers working with the WS RLS. It provides reference information for application developers, including APIs, architecture, procedures for using the APIs and code samples.

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. WS Replica Location Service (WS RLS) Security Considerations	2
2. Usage scenarios	3
3. Tutorials	4
4. Architecture and design overview	5
5. APIs	6
1. Programming Model Overview	6
2. Component API	6
6. WS-Resources and WSDL	7
1. WS RLS Resource Properties	7
I. WS RLS Commands	?
globus-repicalocation-createmappings	9
globus-repicalocation-addmappings	10
globus-repicalocation-deletemappings	11
globus-repicalocation-defineattributes	12
globus-repicalocation-undefineattributes	13
globus-repicalocation-addattributes	14
globus-repicalocation-modifyattributes	15
globus-repicalocation-removeattributes	16
7. Custom query dialect	17
8. Configuring WS RLS	18
1. Configuration overview	18
2. WS RLS Web Service Deployment Descriptor (WSDD) Configuration	18
3. WS RLS Java Naming & Directory Interface (JNDI) Configuration	19
9. Debugging	20
1. Development Logging in Java WS Core	20
2. Increasing log verbosity	20
10. Troubleshooting	21
1. Errors	21
11. Related Documentation	22
Index	23

List of Tables

1. globus-replication-createmappings Options	9
2. globus-replication-addmappings Options	10
3. globus-replication-deletemappings Options	11
4. globus-replication-defineattributes Options	12
5. globus-replication-undefineattributes Options	13
6. globus-replication-addattributes Options	14
7. globus-replication-modifyattributes Options	15
8. globus-replication-removeattributes Options	16
10.1. WS Replica Location Service (WS RLS) Errors	21

Chapter 1. Before you begin

1. Feature summary

Features in GT 4.2.1 WS RLS

- Initial implementation of the WS RLS, which provides a Web services interface to the existing RLS.
- WS-ReplicaLocationCatalog: WSDL-defined interface for performing LRC operations, including creating/adding/deleting mappings, defining/undefining attribute definitions, and creating/modifying/removing attributes. The interface also defines a conceptual set of WS-ResourceProperties based upon which queries may be performed.
- A simple query dialect which may be used to form expressions to be used to query the WS-RLS based on logical/target names and attributes.
- WS-ReplicaLocationIndex: WSDL-defined interface for performing RLI operations. Since the RLI is used as a read-only query interface (querying the index of logical name entries), the WSDL defines domain specific operations. The operation of interest is the standard QueryResourceProperties.

2. Tested platforms

Tested platforms include:

- Linux (RedHat)

3. Backward compatibility summary

Protocol changes since GT 4.0.x

- None.

API changes since GT 4.0.x

- None.

Exception changes since GT 4.0.x

- None.

Schema changes since GT 4.0.x

- None.

4. Technology dependencies

WS RLS depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Replica Location Service (RLS)

WS RLS depends on the following 3rd party software:

- None

5. WS Replica Location Service (WS RLS) Security Considerations

Security recommendations include:

- The following paragraph describes our INITIAL IMPLEMENTATION in this first look of the WS RLS interface -- THIS IS NOT INTENDED TO BE A FINAL SOLUTION.

Users of the WS RLS authenticate themselves to the WS RLS in the usual manner based on their credential. The WS RLS uses the WSAA and Authorization Framework to make authorization decisions. Then, the WS RLS uses a single certificate and key file to identify itself to the RLS irregardless of the user accessing the WS RLS. Thus users that can access the WS RLS are given the fixed WS RLS identity to access the RLS. Users accustomed to using RLS may not feel comfortable with this approach -- if it may be an issue for your environment, we suggest that you not use it with your production RLS. If you think of WS RLS as the gating interface to the RLS, as you should, then you should apply the appropriate authorization restrictions at the WS RLS level, which can be done using WSAA. This will in effect achieve a level of authorization similar to that of the RLS.

Chapter 2. Usage scenarios

There are no developer usage scenarios at this time.

Chapter 3. Tutorials

There are no tutorials available at this time.

Chapter 4. Architecture and design overview

The WS RLS is deployed to the Java WS Core service container. It connects to a RLS service to perform the replica location operations. It serves as an alternative front-end to the current RLS interfaces. See [Replica Location Service \(RLS\)](#) architecture for more information.

Chapter 5. APIs

1. Programming Model Overview

The WS RLS is a WS-RF compliant service implemented using the Globus Java WS Core. It exposes a set of Resource Properties and operations to allow users to create, add and remove mappings from logical names to target names, to define and undefine attribute definitions associated with names, and to create, modify and delete attributes associated with names. A custom query expression dialect is defined by the WS RLS which may be used to query the mapping and attributes represented by the Resource Properties of the service.

2. Component API

The WSDL defines the only public interface to the WS RLS at this time.

Chapter 6. WS-Resources and WSDL

The WS RLS is comprised of two WS-Resources, the `ReplicaLocationCatalog` and the `ReplicaLocationIndex`. Each WS-Resource defines `ResourceProperties` and a set of operations. The `ReplicaLocationCatalog` (or "catalog") defines the set of operations for manipulating name mappings and associated attributes along with the standard query resource properties operation. The `ReplicaLocationIndex` (or "index") specifies only the query resource properties operation since it is a read-only interface to the underlying index. Lifetime operations (*set lifetime* and *destroy*) do not apply to the WS RLS, because its WS-Resources live indefinitely. The resources are available so long as the service container is running and the WS RLS is deployed within the service container. Addressing the resources requires only the URL portion of the WS-Address.

- For detailed information, please see the [schema files](#)¹.
- For the specification of types see [Replica Location Types XSD](#)².
- For the specification of port types see [Catalog Port Type WSDL](#)³ and [Index Port Type WSDL](#)⁴.

1. WS RLS Resource Properties

The resource properties for the `ReplicaLocationCatalog` `ReplicaLocationIndex` port types are listed below:



Note

As a *preview* component the current WS-RLS specifies the following RPs but in most cases does not implement them in the current release. The developers of the component are providing the interfaces for review.

1.1. ReplicaLocationCatalog Resource Properties

- `configuration`: A listing of the configuration settings for the underlying RLS service.
- `diagnostics`: A listing of diagnostics (e.g., status) from the underlying RLS service.
- `catalog`: A resource property style representation of the catalog contexts (i.e., attribute definitions, attributes, and mappings) of the underlying RLS service.

1.2. ReplicaLocationIndex Resource Properties

- `configuration`: A listing of the configuration settings for the underlying RLS service.
- `diagnostics`: A listing of diagnostics (e.g., status) from the underlying RLS service.
- `catalog`: A resource property style representation of the index contexts (i.e., attribute definitions, attributes, and mappings) of the underlying RLS service.

¹ http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/common/schema/replica/location/?only_with_tag=HEAD

² http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/common/schema/replica/location/replicolocation_types.xsd?rev=1.2&only_with_tag=HEAD&content-type=text/vnd.viewcvs-markup

³ http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/common/schema/replica/location/catalog_port_type.wsdl?rev=1.1&only_with_tag=HEAD&content-type=text/vnd.viewcvs-markup

⁴ http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/common/schema/replica/location/index_port_type.wsdl?rev=1.1&only_with_tag=HEAD&content-type=text/vnd.viewcvs-markup

WS RLS Commands

The WS RLS provides a set of command-line tools to create, add, remove mappings between logical names and target names, define and undefine attribute definitions, and create, modify, and delete attributes. These command line tools are available on Unix and Windows platforms and will work in the same way (of course within the platform rules - the path syntax, variable definitions, etc.).

The WS RLS command-line tools make use of the Common Java Client Options. These options are referred to below as [options].

Name

`globus-replication-createmappings` -- This tool is used to create mappings between logical names and target names. The *create* semantic implies that the logical name does not exist at the time of invocation.

`globus-replication-createmappings`

Tool description

Use this tool to create mappings between logical names and target names in the replica location catalog. The mapping must not exist. In addition, the logical name must not exist.

Command syntax

```
globus-replication-createmappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 1. `globus-replication-createmappings` Options

<code>{ logical-name target-name }+</code>	A listing of logical name to target name mappings.
<code>input-file</code>	A file containing logical name to target name mappings.
<code>-</code>	Standard input stream containing logical name to target name mappings.

Name

`globus-repicalocation-addmappings` -- This tool is used to add mappings between logical names and target names. The *add* semantic implies that the logical name does exist at the time of invocation.

`globus-repicalocation-addmappings`

Tool description

Use this tool to add mappings between logical names and target names in the replica location catalog. The mapping must not exist. In addition, the logical name must exist.

Command syntax

```
globus-repicalocation-addmappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 2. `globus-repicalocation-addmappings` Options

<code>{ logical-name target-name }+</code>	A listing of logical name to target name mappings.
<code>input-file</code>	A file containing logical name to target name mappings.
<code>-</code>	Standard input stream containing logical name to target name mappings.

Name

globus-repicalocation-deletemappings -- This tool is used to delete mappings between logical names and target names.

globus-repicalocation-deletemappings

Tool description

Use this tool to delete mappings between logical names and target names in the replica location catalog. The mapping must exist.

Command syntax

```
globus-repicalocation-deletemappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 3. globus-repicalocation-deletemappings Options

{ logical-name target-name }+	A listing of logical name to target name mappings.
input-file	A file containing logical name to target name mappings.
-	Standard input stream containing logical name to target name mappings.

Name

globus-replication-defineattributes -- This tool is used to define attributes.

globus-replication-defineattributes

Tool description

Use this tool to define attributes. Attribute definitions must be given a name unique within the local instance of the replica location catalog. Attribute definitions must be given a value type of dateTime, decimal, integer, or string. And attribute definitions must be associated with an object type of logical or target.

Command syntax

```
globus-replication-defineattributes [options] \  
{ { name object-type value-type }+ | input-file | - }
```

Table 4. globus-replication-defineattributes Options

{ name object-type value-type }+	A listing of attribute name, associated object-type, and value-type.
input-file	A file containing the listing of attribute name, associated object-type, and value-type.
-	Standard input stream containing the listing of attribute name, associated object-type, and value-type.

Name

globus-repicalocation-undefineattributes -- This tool is used to undefine attributes.

globus-repicalocation-undefineattributes

Tool description

Use this tool to undefine attributes. Attribute definitions must be identified by the definition's name and associated object-type. The operation will clear attribute values for existing attributes with the definition's name.

Command syntax

```
globus-repicalocation-undefineattributes [options] \  
{ { name object-type }+ | input-file | - }
```

Table 5. globus-repicalocation-undefineattributes Options

{ name object-type }+	A listing of attribute name and associated object-type.
input-file	A file containing the listing of attribute name and associated object-type.
-	Standard input stream containing the listing of attribute name and associated object-type.

Name

globus-repicalocation-addattributes -- This tool is used to add attributes.

globus-repicalocation-addattributes

Tool description

Use this tool to add attributes associated with logical names or target names. A corresponding attribute definition must exist. The logical name or target name with which to associate the attribute must exist. There must not be an existing attribute of the same type for a given logical name or target name. When adding attributes, the following parameters are required. The logical name or target name, referred to as the key. The name of the attribute as defined by an existing attribute definition. An object-type of logical or target. A value-type corresponding to dateTime, decimal, integer, or string. And finally a value compatible with the value-type.

Command syntax

```
globus-repicalocation-addattributes [options] \  
{ { key name object-type value-type value }+ | input-file | - }
```

Table 6. globus-repicalocation-addattributes Options

{ key name object-type value-type value }+	A listing of key, attribute name, associated object-type, value-type, and value.
input-file	A file containing the listing of key, attribute name, associated object-type, value-type, and value.
-	Standard input stream containing the listing of key, attribute name, associated object-type, value-type, and value.

Name

globus-replication-modifyattributes -- This tool is used to modify attributes.

globus-replication-modifyattributes

Tool description

Use this tool to modify attributes associated with logical names or target names. Mutability of attributes is limited only to the attribute's value. The corresponding attribute must exist.

Command syntax

```
globus-replication-modifyattributes [options] \  
{ { key name object-type value-type value }+ | input-file | - }
```

Table 7. globus-replication-modifyattributes Options

{ key name object-type value-type value }+	A listing of key, attribute name, associated object-type, value-type, and value.
input-file	A file containing the listing of key, attribute name, associated object-type, value-type, and value.
-	Standard input stream containing the listing of key, attribute name, associated object-type, value-type, and value.

Name

globus-repicalocation-removeattributes -- This tool is used to remove existing attributes.

globus-repicalocation-removeattributes

Tool description

Use this tool to remove existing attributes associated with logical names or target names. The corresponding attribute must exist.

Command syntax

```
globus-repicalocation-removeattributes [options] \  
{ { key name object-type }+ | input-file | - }
```

Table 8. globus-repicalocation-removeattributes Options

{ key name object-type }+	A listing of key, attribute name, and associated object-type.
input-file	A file containing the listing of key, attribute name, and associated object-type.
-	Standard input stream containing the listing of key, attribute name, and associated object-type.

Chapter 7. Custom query dialect

The WS RLS defines a custom query dialect and provides a custom expression evaluator to support the dialect. The dialect is noticeably rudimentary. We expect to improve the dialect as the WS RLS matures. The following list enumerates the possible expressions when using the custom "<http://globus.org/replica/location/06/01/QueryDialect>" dialect provided by the WS RLS.

- query-target: {logical-name}+
Queries the catalog based on logical names and returns matching target names.
- query-logical: {target-name}+
Queries the catalog based on target names and returns matching logical names.
- query-index-logical: {logical-name}+
Queries the index based on logical names and returns matching target names.
- exists-logical: {logical-name}+
Evaluates existence of logical name in the catalog.
- exists-target: {target-name}+
Evaluates existence of target name in the catalog.
- query-logical-attributes: {logical-name}+
Queries the catalog for attributes associated with logical names.
- query-logical-named-attributes: attribute-name {logical-name}+
Queries the catalog for attributes based on the given *attribute-name* for the associated logical names.
- query-target-attributes: {target-name}+
Queries the catalog for attributes associated with target names.
- query-target-named-attributes: attribute-name {target-name}+
Queries the catalog for attributes based on the given *attribute-name* for the associated target names.

Chapter 8. Configuring WS RLS

1. Configuration overview

The WS RLS requires certain WSDD and JNDI settings to be properly configured. The installed JNDI configuration file may be found at `$GLOBUS_LOCATION/etc/globus_wsrf_replicolocation_resource/jndi-config.xml`. To view the default JNDI file (current snapshot) from the Globus CVS repository [click here](#)¹. The installed WSDD configuration file may be found at `$GLOBUS_LOCATION/etc/globus_wsrf_replicolocation_service/server-config.wsdd`. To view the default WSDD file (current snapshot) from the Globus CVS repository [click here](#)². In most cases, you will not need to alter either of these files.

This information is in addition to the basic configuration instructions in the [Installing GT 4.2.1](#).

2. WS RLS Web Service Deployment Descriptor (WSDD) Configuration

The WS RLS's WSDD determines which services will be activated by the GT Java WS Core container. By default, the WS RLS defines two services.

- `ReplicaLocationCatalogService`. The `ReplicaLocationCatalogService` service provides the WS interface to the RLS Local Replica Catalog capabilities. If you intend to activate this service, you need not alter the WSDD. If you do not wish to activate this service, simply remove it from the WSDD and restart the container.

```
...
    <service name="ReplicaLocationCatalogService" provider="Handler"
        use="literal" style="document">
...
</service>
...
```

- `ReplicaLocationIndexService`. The `ReplicaLocationIndexService` service provides the WS interface to the RLS Replica Location Index capabilities. If you intend to activate this service, you need not alter the WSDD. If you do not wish to activate this service, simply remove it from the WSDD and restart the container.

```
...
    <service name="ReplicaLocationIndexService" provider="Handler"
        use="literal" style="document">
...
</service>
...
```

¹ <http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/resource/java/source/deploy-jndi-config.xml>

² <http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/service/java/source/deploy-server.wsdd>

3. WS RLS Java Naming & Directory Interface (JNDI) Configuration

The WS RLS's JNDI configuration determines the resource class to be used for each service defined by the WS RLS WSDD. It also determines other critical settings used by these resources. In this file, you will find settings for the two default services defined in the WSDD.

Each service has a JNDI Resource of name `RLSConnectionSource`. This JNDI Resource is used by the WS RLS service to open a connection to the backend RLS service. In many cases, you may find that the default settings require no modification for your setup. The `url` parameter sets the endpoint for the backend RLS. The `certfile` and `keyfile` parameters set the certificate and key files to be used by the WS RLS to communicate securely with the backend RLS.

You will notice that the default settings use the container's suggested certificate and key file paths. If you have chosen different paths in which to keep these files, please adjust the settings accordingly. At present, the WS RLS uses a single identity (defined by the certificate and key) to authenticate itself with the RLS in order to perform any user requested operations. When the WS RLS interface is finalized, it will most likely use a different approach to authentication between WS RLS and the back end RLS. At this time, we expect that the interface will change such that the caller's credentials will be used to make calls between the WS RLS and the back end RLS on behalf of the originating caller.

```
...
  <resource
    name="RLSConnectionSource"
    type="org.globus.replica.rls.connection.impl.PooledRLSConnectionSource">
  <resourceParams>
    <parameter>
      <name>factory</name>
      <value>org.globus.wsrfl.jndi.BeanFactory</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>rls://localhost</value>
    </parameter>
    <parameter>
      <name>certfile</name>
      <value>/etc/grid-security/containercert.pem</value>
    </parameter>
    <parameter>
      <name>keyfile</name>
      <value>/etc/grid-security/containerkey.pem</value>
    </parameter>
  </resourceParams>
</resource>
...
```

Chapter 9. Debugging

Because WS-RLS is built on top of Java WS Core, developer debugging is the same as described in [Chapter 10, Debugging](#). For sys admin debugging information, see [Chapter 5, Debugging](#).

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](#)¹ 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](#)² as a logging implementation. Log4j uses a separate configuration file to configure itself. Please see Log4j documentation for details on the [configuration file format](#)³.

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](#)⁴. 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
```

1.2. 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.

2. Increasing log verbosity

Generating verbose log output is a critical aid in troubleshooting of the WS RLS and is useful when communicating problems to Globus support lists. To increase logging detail, add the following lines to the `$GLOBUS_LOCATION/container-log4j.properties` file.

```
...
log4j.category.org.globus.replica.location=DEBUG
log4j.category.org.globus.replica.rls=DEBUG
...
```

¹ <http://jakarta.apache.org/commons/logging/>

² <http://logging.apache.org/log4j/>

³ [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))

⁴ <http://tomcat.apache.org/tomcat-5.5-doc/logging.html>

Chapter 10. Troubleshooting

For sys admin debugging information, see [Chapter 5, Debugging](#).

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

1. Errors

Table 10.1. WS Replica Location Service (WS RLS) Errors

Error Code	Definition	Possible Solutions
Error: java.lang.Null- PointerException	When invoking the WS RLS command-line clients, a system-level exception like the one above may be encountered.	The admin should check the container logs for the exact error.
Error: A server error occurred while processing the request	When invoking the WS RLS command-line clients, a server error like the one above may be encountered.	The admin should check the container logs for the exact error.
java.lang.UnsatisfiedLinkError	This exception when using the WS RLS may indicate that the native RLS libraries that WS RLS depends on could not be located.	To correct this problem, ensure that the \$GLOBUS_LOCATION/lib directory is in the library search path (on some systems this is the LD_LIBRARY_PATH variable).
Unable to connect to localhost:39281	The WS RLS is an interface layer that depends on the RLS for the replica location functionality. You must install and run RLS and configure WS RLS to use the RLS via its JNDI configuration.	Check that RLS is installed, running, and check that the WS RLS JNDI configuration uses the correct host-name and port to connect to the RLS.
org.globus.com-mon.ChainedIOException: Failed to initialize security context	If this exception occurs while using WS RLS, it may indicate that the user's proxy is invalid.	To correct the error, the user must properly initialize the user proxy. See grid-proxy-init for more information on proxy initialization.
Error: org.xml.sax.SAXException: Unregistered type: class xxx	If this exception occurs when using the WS RLS, it may indicate that an Axis generated XML type, defined by the WS RLS XSD, was not properly registered. While all the XML types should get registered upon deployment without intervention by the user, sometimes they do not.	To remedy the situation add a typeMapping to the server-config.wsdd file under globus_wsrf_replicalocation_service. Use the format shown here .

Chapter 11. Related Documentation

Not available at this time.

Index

A

adding attributes, 14
adding mappings, 10

C

configuring ReplicaLocationCatalogService, 18
configuring ReplicaLocationIndexService, 18
configuring WS RLS, 18
creating mappings, 9

D

debugging
 (for developers), 20
 logging, 20
defining attributes, 12
deleting mappings, 11

E

errors, 21

L

logging, 20
 debugging, 20

M

modifying attributes, 15

Q

querying (using custom query dialect), 17

R

removing attributes, 16

T

troubleshooting
 errors, 21

U

undefining attributes, 13

GT 4.2.1 Migrating Guide for WS RLS

Table of Contents

1. Migrating from GT4.0	1
2. Migrating from GT3	1
3. Migrating from GT2	1

<titleabbrev>Migrating Guide</titleabbrev>

The following provides available information about migrating from previous versions of the Globus Toolkit.

1. Migrating from GT4.0

The public interfaces to Java WS Core (on which WS RLS is based) have changed and are not backwards compatible. For information on updating your services, see [Section 1, “Updating higher level services”](#).

2. Migrating from GT3

None

3. Migrating from GT2

None

GT 4.2.1 WS RLS: Quality Profile

Table of Contents

- 1. Test coverage reports 1
- 2. Code analysis reports 1
- 3. Outstanding bugs 1
- 4. Bug Fixes 1
- 5. Performance reports 1

<titleabbrev>Quality Profile</titleabbrev>

1. Test coverage reports

- TBD

2. Code analysis reports

- TBD

3. Outstanding bugs

- None

4. Bug Fixes

- None

5. Performance reports

- TBD

GT 4.2.1 Release Notes: WS RLS

Table of Contents

1. Component Overview	1
2. Feature summary	1
3. Summary of Changes in WS RLS	2
4. Bug Fixes	2
5. Known Problems	2
6. Technology dependencies	2
7. Tested platforms	2
8. Backward compatibility summary	3
9. Associated Standards	3
10. For More Information	3

<titleabbrev>Release Note</titleabbrev>

1. Component Overview

The WS Replica Location Service (WS RLS) provides a Web services interface for the existing [Replica Location Service \(RLS\)](#). The features of the RLS have been preserved and interfaces should look familiar to users accustomed to working with the RLS. In most cases the interfaces are semantically similar (or identical) to the RLS operations, but effort has been made to adapt the interfaces to conform to expected style and conventions more typical of Web services. The interfaces consist of the WS-ReplicaLocationCatalog and the WS-ReplicaLocationIndex, representing the RLS Local Replica Catalog (LRC) and RLS Replica Location Index (RLI).



Note

Please note this is relatively new code and may not necessarily be as robust as the rest of RLS.

2. Feature summary

Features in GT 4.2.1 WS RLS

- Initial implementation of the WS RLS, which provides a Web services interface to the existing RLS.
- WS-ReplicaLocationCatalog: WSDL-defined interface for performing LRC operations, including creating/adding/deleting mappings, defining/undefining attribute definitions, and creating/modifying/removing attributes. The interface also defines a conceptual set of WS-ResourceProperties based upon which queries may be performed.
- A simple query dialect which may be used to form expressions to be used to query the WS-RLS based on logical/target names and attributes.
- WS-ReplicaLocationIndex: WSDL-defined interface for performing RLI operations. Since the RLI is used as a read-only query interface (querying the index of logical name entries), the WSDL defines domain specific operations. The operation of interest is the standard QueryResourceProperties.

3. Summary of Changes in WS RLS

No changes since GT 4.2.0.

4. Bug Fixes

- None

5. Known Problems

The following problems and limitations are known to exist for WS RLS at the time of the 4.2.1 release:

5.1. Limitations

- At this time, WS RLS only supports a limited query response for WS-ReplicaLocationIndex (i.e., index) lookups. When querying the index for a given logical name (say 'my-logical-name-x1105') the response is the URL for the catalogs that contain entries corresponding to the logical name (say 'https://mycataloghost:8443'). The limitation to this interface is that the complete URL path is not included in the response. Only the scheme, hostname, and port of the URL are returned to the client. The client must append the remaining path to form the complete URL suitable for contacting the catalog (in this example 'https://mycataloghost:8443/wsrf/services/ReplicaLocationCatalogService').
- At this time, dateTime attribute values are not supported. If you are interested in this feature, please contact us.
- At this time, the command-line tools for the WS RLS do not support the documented '-' option, which is intended to allow for command-line parameters to be read from `stdin`.

5.2. Outstanding bugs

- None

6. Technology dependencies

WS RLS depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Replica Location Service (RLS)

WS RLS depends on the following 3rd party software:

- None

7. Tested platforms

Tested platforms include:

- Linux (RedHat)

8. Backward compatibility summary

Protocol changes since GT 4.0.x

- None.

API changes since GT 4.0.x

- None.

Exception changes since GT 4.0.x

- None.

Schema changes since GT 4.0.x

- None.

9. Associated Standards

Associated standards for WS RLS:

- Conforms to WS-RF family of specifications and WS-Addressing and WS-Security.

10. For More Information

See [WS Replica Location Service \(WS RLS\)](#) for more information about this component.