

GT 4.2.1 Component Guide to Public Interfaces: WS RLS

GT 4.2.1 Component Guide to Public Interfaces: WS RLS

Table of Contents

1. APIs	1
1. Programming Model Overview	1
2. Component API	1
2. WS-Resources and WSDL	2
1. WS RLS Resource Properties	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
3. Custom query dialect	12
4. Configuring WS RLS	13
1. Configuration overview	13
2. WS RLS Web Service Deployment Descriptor (WSDD) Configuration	13
3. WS RLS Java Naming & Directory Interface (JNDI) Configuration	14
A. Errors	15

List of Tables

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

Chapter 1. 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 2. 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 ReplicaLocatoinIndex 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 sytle 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 sytle 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-repicalocation-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-repicalocation-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-repicalocation-createmappings [options] \  
{ { logical-name target-name }+ | input-file | - }
```

Table 1. globus-repicalocation-createmappings 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-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 3. 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 4. 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 `$GLOBUS_LOCATION/etc/globus_wsrf_replicolocation_resource/jndi-config.xml`. To view the default JNDI file (current snapshot) from the Globus CVS repository [click here](http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/resource/java/source/deploy-jndi-config.xml)¹. The installed WSDD configuration file may be found at `$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](http://viewcvs.globus.org/viewcvs.cgi/ws-replica/location/service/java/source/deploy-server.wsdd)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>
...
```

Appendix A. Errors

Table A.1. WS Replica Location Service (WS RLS) Errors

Error Code	Definition	Possible Solutions
Error: java.lang.NullPointerException	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.common.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_replica_location_service. Use the format shown here .