GT 5.0.4 Component Guide to Public Interfaces: RLS
GT 5.0.4 Component Guide to Public Interfaces: RLS
# Table of Contents

1. APIs .............................................................................................................................................. 1  
   1. Programming Model Overview .......................................................................................... 1  
   2. Component API ................................................................................................................. 1  
2. Command line tools .................................................................................................................... 2  
   globus-rls-admin ................................................................................................................... 3  
   globus-rls-cli ......................................................................................................................... 5  
   globus-rls-server .................................................................................................................. 9  
3. Configuring RLS .......................................................................................................................... 14  
   1. Configuration overview ................................................................................................. 14  
   2. Server configuration file (globus-rls-server.conf) ......................................................... 14  
   3. Basic configuration ......................................................................................................... 14  
   4. Host key and certificate configuration ............................................................................. 15  
   5. Configuring LRC to RLI updates ..................................................................................... 16  
   6. Configuring the RLS Server for the MDS2 GRIS ............................................................ 17  
   7. Complete RLS Server settings (globus-rls-server.conf) ................................................ 17  
A. Errors .......................................................................................................................................... 23  
Glossary ......................................................................................................................................... 24
List of Tables

2.1. Options for globus-rls-admin ............................................................................................................ 4
2.2. Options for globus-rls-cli ................................................................................................................. 5
2.3. Commands for globus-rls-cli ............................................................................................................ 7
2.4. Options for globus-rls-server .......................................................................................................... 12
3.1. Complete RLS Server settings (globus-rls-server.conf) ................................................................. 18
A.1. Replica Locator Service (RLS) Errors ............................................................................................. 23
Chapter 1. APIs

1. Programming Model Overview

The RLS provides a Client API for C based clients. The RLS Client C API is provided in the form of a library (e.g., .so file). Any installation of RLS will include the shared library as part of the $GLOBUS_LOCATION/include and $GLOBUS_LOCATION/lib directories.

2. Component API

- RLS Client C API Documentation

\[^1\]

\[^1\] http://www.isi.edu/~annc/rls/doc/client/index.html
Chapter 2. Command line tools
Name
globus-rls-admin -- RLS administration tool
globus-rls-admin

Tool description
Performs administrative operations on an RLS server.

Synopsis
-A|a|C option value|-c option|-d|-e|-p|-q|-s|-t timeout|-u|-v [ rli ] [ pattern ] [ server ]
Options

Table 2.1. Options for globus-rls-admin

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| -A     | Adds rli to the list of RLI servers updated by an LRC server using Bloom filters.  

*Note:* Partitions are not supported with Bloom filters. The LRC server maintains one Bloom filter for all LFNs in its database, which is sent to all RLI servers configured to receive Bloom filter updates with this option. |
| -a     | Adds rli and optionally pattern to the list of RLI servers that the LRC server sends updates to (using a list of LFNs).  

If pattern is specified, then only LFNs matching it will be sent to rli.  

If rli is added with no patterns, then it is sent all updates. Pattern matching is done using standard Unix file globbing. |
| -C option value | Sets server option to value.  

*Important:* This does not update the configuration file. The next time the server is restarted, the configuration change will be lost. |
| -c option | Retrieves the configuration value for the specified option from the server.  

If option is set to all, then all options are retrieved. |
| -d     | Removes rli and pattern from the list of RLI servers that the LRC server sends updates to.  

If pattern is not specified, then all entries for rli are removed.  

*Note:* If all patterns are removed separately, then rli is sent all updates. To stop any updates from being sent to rli, do not specify pattern. |
| -e     | Clears the LRC database. Removes all lfn, pfn mappings. |
| -p     | Verifies that the server is responding. |
| -q     | Causes the RLS server to exit. |
| -S     | Shows statistics and other information gathered by the RLS server.  

This is intended to be input into GRIS. |
| -s     | Shows the list of RLI servers and patterns being sent updates by the LRC server.  

If rli or pattern are not specified, they are considered wildcards. |
| -t timeout | Sets timeout (in seconds) for RLS server requests.  

The default value is 30. |
| -u     | Causes the LRC server to immediately start full soft state updates to any RLI servers previously added with the -a option. |
| -v     | Shows the version and exits. |
Name
globus-rls-cli -- RLS client tool
globus-rls-cli

Tool description

Provides a command line interface to some of the functions supported by RLS. It also supports an interactive interface (if command is not specified). In interactive mode, double quotes may be used to encode an argument that contains white space.

Synopsis

command [ -c ] [ -h ] [ -l reslimit ] [ -s ] [ -t timeout ] [ -u ] [ command ] rls-server

Options

The client command tool uses getopt for command line parsing.

Note: Some versions will continue scanning for options (works that begin with a hyphen) for the entire command line, which makes it impossible to specify negative integer or floating point value for an attribute. The workaround for this problem is to tell getopt() that there are no more options by including 2 hyphens. For example, to specify the value -2 you must enter -- -2.

Table 2.2. Options for globus-rls-cli

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c</td>
<td>Sets &quot;clearvalues&quot; flag when deleting an attribute (will remove any attribute value records when an attribute is deleted).</td>
</tr>
<tr>
<td>-h</td>
<td>Shows usage.</td>
</tr>
<tr>
<td>-l reslimit</td>
<td>Sets an incremental limit on the number of results returned by a wildcard query at a time. Note that all results will be returned by the client. This parameter only limits the number of results incrementally retrieved by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user's perspective the client will simply return all 1000 results. Zero means no limit.</td>
</tr>
<tr>
<td>-s</td>
<td>Uses SQL style wildcards (% and _).</td>
</tr>
<tr>
<td>-t timeout</td>
<td>Sets timeout (in seconds) for RLS server requests. The default is 30 seconds.</td>
</tr>
<tr>
<td>-u</td>
<td>Uses Unix style wildcards (* and ?).</td>
</tr>
<tr>
<td>-v</td>
<td>Shows version.</td>
</tr>
</tbody>
</table>
Commands
Table 2.3. Commands for globus-rls-cli

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>add</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code></td>
<td>Adds a pfn to mappings of lfn in an LRC catalog.</td>
</tr>
<tr>
<td><strong>attribute add</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt; &lt;attr-type&gt;</code></td>
<td>Adds an attribute to an object, where object should be the lfn or pfn name. obj-type should be one of lfn or pfn. attr-type should be one of date, float, int, or string. If value is of type date then it should be in the form &quot;YYYY-MM-DD HH:MM:DD&quot;.</td>
</tr>
<tr>
<td><strong>attribute bulk add</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Bulk adds attribute values.</td>
</tr>
<tr>
<td><strong>attribute bulk delete</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Bulk deletes attributes.</td>
</tr>
<tr>
<td><strong>attribute bulk query</strong> <code>&lt;attr&gt; &lt;obj-type&gt; &lt;object&gt;</code></td>
<td>Bulk queries attributes.</td>
</tr>
<tr>
<td><strong>attribute define</strong> <code>&lt;attr&gt; &lt;obj-type&gt; &lt;attr-type&gt;</code></td>
<td>Defines a new attribute.</td>
</tr>
<tr>
<td><strong>attribute delete</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Removes attribute from object.</td>
</tr>
<tr>
<td><strong>attribute modify</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt; &lt;attr-type&gt;</code></td>
<td>Modifies the value of an attribute.</td>
</tr>
<tr>
<td><strong>attribute query</strong> <code>&lt;object&gt; &lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Retrieves the value of the specified attribute for object.</td>
</tr>
<tr>
<td><strong>attribute search</strong> <code>&lt;attr&gt; &lt;obj-type&gt; &lt;operator&gt; &lt;attr-type&gt;</code></td>
<td>Searches for objects which have the specified attribute matching operator and value. operator should be one of =, !=, &gt;, &gt;=, &lt;, or &lt;=.</td>
</tr>
<tr>
<td><strong>attribute show</strong> <code>&lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Shows an attribute definition. If attr is a hyphen (-) then all attributes are shown.</td>
</tr>
<tr>
<td><strong>attribute undefine</strong> <code>&lt;attr&gt; &lt;obj-type&gt;</code></td>
<td>Deletes an attribute definition. Will return an error if any objects possess this attribute.</td>
</tr>
<tr>
<td><strong>bulk add</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code> <code>[&lt;lfn&gt; &lt;pfn&gt;]</code></td>
<td>Bulk adds lfn, pfn mappings.</td>
</tr>
<tr>
<td><strong>bulk create</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code> <code>[&lt;lfn&gt; &lt;pfn&gt;]</code></td>
<td>Bulk creates lfn, pfn mappings.</td>
</tr>
<tr>
<td><strong>bulk delete</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code> <code>[&lt;lfn&gt; &lt;pfn&gt;]</code></td>
<td>Bulk deletes lfn, pfn mappings.</td>
</tr>
<tr>
<td><strong>bulk query lrc lfn</strong> <code>[&lt;lfn&gt; ...]</code></td>
<td>Bulk queries the LRC for lfns.</td>
</tr>
<tr>
<td><strong>bulk query rli lfn</strong> <code>[&lt;lfn&gt; ...]</code></td>
<td>Bulk queries the RLI for lfns.</td>
</tr>
<tr>
<td><strong>create</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code></td>
<td>Creates a new lfn, pfn mapping in an LRC catalog.</td>
</tr>
<tr>
<td><strong>delete</strong> <code>&lt;lfn&gt; &lt;pfn&gt;</code></td>
<td>Deletes a lfn, pfn mapping from an LRC catalog.</td>
</tr>
<tr>
<td><strong>exit</strong></td>
<td>Exits the interactive session.</td>
</tr>
<tr>
<td><strong>help</strong></td>
<td>Prints a help message.</td>
</tr>
<tr>
<td><strong>query lrc lfn</strong> <code>&lt;lfn&gt;</code></td>
<td>Queries an LRC server for mappings of lfn.</td>
</tr>
<tr>
<td><strong>query lrc pfn</strong> <code>&lt;pfn&gt;</code></td>
<td>Queries an LRC server for mappings to pfn.</td>
</tr>
<tr>
<td><strong>query rli lfn</strong> <code>&lt;lfn&gt;</code></td>
<td>Queries an RLI server for mappings of lfn.</td>
</tr>
<tr>
<td>Command line tools</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td></td>
</tr>
<tr>
<td><strong>query wildcard lrc lfn &lt;lfn-pattern&gt;</strong></td>
<td>Performs a wildcarded query of an LRC server for mappings of <em>lfn-pattern</em>. Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.</td>
</tr>
<tr>
<td><strong>query wildcard lrc pfn &lt;pfn-pattern&gt;</strong></td>
<td>Queries an LRC server for mappings to <em>pfn-pattern</em>. Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.</td>
</tr>
<tr>
<td><strong>query wildcard rli lfn &lt;lfn-pattern&gt;</strong></td>
<td>Queries an RLI server for mappings of <em>lfn-pattern</em>. Patterns use the standard Unix wildcard characters: an asterisk (*) matches 0 or more characters, and a question mark (?) matches any single character.</td>
</tr>
<tr>
<td><strong>set reslimit &lt;limit&gt;</strong></td>
<td>Sets an incremental limit on the number of results returned by a wildcard query at a time. Note that <em>all results will be returned</em> by the client. This parameter only limits the number of results <em>incrementally retrieved</em> by the client during a single internal communication call. For instance, if the wildcard query produces 1000 results and the reslimit is set to 100, the client will internally make 10 calls to the server. From the user’s perspective the client will simply return all 1000 results.</td>
</tr>
<tr>
<td><strong>set timeout &lt;timeout&gt;</strong></td>
<td>Sets the timeout (in seconds) on calls to the RLS server. The default value is <strong>30</strong>.</td>
</tr>
<tr>
<td><strong>version</strong></td>
<td>Shows the version and exits.</td>
</tr>
</tbody>
</table>
Name

globus-rls-server -- RLS server tool

globus-rls-server

Tool description

The RLS server (globus-rls-server) can be configured as either one or both of the following:

- **Location Replica Catalog (LRC)** server, which manages *Logical File Name* (LFN) to *Physical File Name* (PFN) mappings in a database. *Note:* If globus-rls-server is configured as an LRC server, the RLI servers that it sends updates to should be added to the database using globus-rls-admin.

- **Replica Location Index (RLI)** server, which manages mappings of LFNs to LRC servers.

Clients wishing to locate one or more physical filenames associated with a logical filename should first contact an RLI server, which will return a list of LRCs that may know about the LFN. The LRC servers are then contacted in turn to find the physical filenames.

*Note:* RLI information may be out of date, so clients should be prepared to get a negative response when contacting an LRC (or no response at all if the LRC server is unavailable).

Synopsis

```
```

LRC to RLI Updates

Two methods exist for LRC servers to inform RLI servers of their LFNs.

- By default, the LFNs are sent from the LRC to the RLI. This can be time consuming if the number of LFNs is large, but it does give the RLI an exact list of the LFNs known to the LRC, and it allows wildcard searching of the RLI.

- Alternatively, *Bloom filters* may be sent, which are highly compressed summaries of the LFNs. However, they do not allow wildcard searching and will generate more "false positives" when querying an RLI.

Please see below for more on Bloom filters.

globus-rls-admin can be used to manage the list of RLIs that an LRC server updates. This includes partitioning LFNs among multiple RLI servers.

A softstate algorithm is used for updates, periodically the source server sends its state (LFN information) to the RLI servers it updates. The RLI servers add these LFNs to their index, or update a timestamp if the LFNs were already known. RLI servers expire information about LFN,LRC mappings if they haven't been updated for a period longer than the softstate update interval.

Options that can be configured to control the softstate algorithm when a source server updates an RLI by sending LFNs include:
• **rli_expire_int (seconds)**
  How often an RLI server will check for stale entries in its database.

• **rli_expire_stale (seconds)**
  How old an entry must be in an RLI database before it's considered stale. This value should be no smaller than `update_ll_int`. Note if the LRC server is responding this value is not used, instead the value of `update_ll_int` or `update_bf_int` is retrieved from the LRC server, multiplied by 1.2, and used as the value for `rli_expire_stale`.

• **update_bf_int (seconds)**
  Interval between RLI updates when using Bloom filters.

• **update_ll_int (seconds)**
  Interval between RLI updates when using LFN lists for softstate updates.

Updates to an LRC (new LFNs or deleted LFNs) normally don't propagate to RLI servers until the next softstate update (controlled by `update_ll_int` and `update_bf_int`). However by enabling "immediate update" mode an LRC will send updates to an RLI within `update_buftime` seconds. Immediate updates are enabled by setting `update_immediate` to true. If updates are done with LFN lists then only the LFNs that have been added or deleted to the source server are sent, if Bloom filters are used then the entire Bloom filter is sent.

When immediate updates are enabled, the interval between softstate updates is multiplied by `update_factor`, so long as no updates have failed (source and RLI are considered to be in sync). This can greatly reduce the number of softstate updates a source needs to send to an RLI. Incremental updates are buffered by the source server until either 100 updates have accumulated (when LFN lists are used), or `update_buftime` seconds have passed since the last update.

### Bloom filter updates

A Bloom filter is an array of bits. Each LFN is hashed multiple times and the corresponding bits in the Bloom filter are set.

Querying an RLI to verify if an LFN exists is done by performing the same hashes and checking if the bits in the filter are on. If not, then the LFN is known not to exist. If they're all on, then all that's known is that the LFN probably exists.

The size of the Bloom filter (as a multiple of the number of LFNs) and the number of hash functions control the false positive rate. The default values of 10 and 3 give a false positive rate of approximately 1%.

The advantage of Bloom filters is their efficiency. For example, if the LRC has 1,000,000 LFNs in its database, with an average length of 20 bytes, then 20,000,000 bytes must be sent to an RLI during a soft state update (assuming no partitioning). The RLI server must perform 1,000,000 updates to its database to create new LFN, LRC mappings or update timestamps on existing entries. With Bloom filters only 1,250,000 bytes are sent (10 x 1,000,000 bits / 8), and there are no database operations on the RLI (Bloom filters are maintained entirely in memory). A comparison of the time to perform a 1,000,000 LFN update: it took 20 minutes sending all the LFNs and less than 1 second using a Bloom filter. However as noted before, Bloom filters do not support wild card searches of an RLI.

**Note:** An LRC server can update some RLIs with Bloom filters and others with LFNs. However, an RLI server can only be updated using one method.

The following options in the Configuration file control Bloom filter updates:

• **rli_bloomfilter true|false**
  RLI servers must have this set to accept Bloom filter updates.
• **rli_bloomfilter_dir none|default|pathname**
  Bloom filters saved in this directory and read at start time if not "none". See CONFIGURATION for details.

• **lrc_bloomfilter_numhash N**
  Number of hash functions, an integer from 1 to 8. The default is 3.

• **lrc_bloomfilter_ratio N**
  Size of the Bloom filter as a multiple of the number of LFNs in the LRC database. Too small a value will generate too many false positives, too large wastes memory and network bandwidth.

Note: An LRC server can update some RLIs with Bloom filters, and others with LFNs. However an RLI server can only be updated using one method, and an RLI acting as a source for updates can only send the type of updates that it receives.

**Log Messages**

**globus-rls-server** uses syslog to log errors and other information (facility LOG_DAEMON) when it's running in normal (daemon) mode.

If the `-d` option (debug) is specified, then log messages are written to stdout.

**Signals**

The server will reread its configuration file if it receives a **HUP** signal. It will wait for all current requests to complete and shut down cleanly if sent any of the following signals: **INT**, **QUIT** or **TERM**.

**Options (globus-rls-server)**

The following table describes the command line options available for globus-rls-server:
### Table 2.4. Options for globus-rls-server

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-B update_bf_int</code></td>
<td>Interval between RLI updates when using Bloom filters.</td>
</tr>
<tr>
<td><code>-b maxbackoff</code></td>
<td>Maximum time (in seconds) that <code>globus-rls-server</code> will attempt to reopen the socket it listens on after an I/O error.</td>
</tr>
<tr>
<td><code>-C rlscertfile</code></td>
<td>Name of the X.509 certificate file that identifies the server; sets environment variable <code>X509_USER_CERT</code>.</td>
</tr>
<tr>
<td><code>-c conffile</code></td>
<td>Name of the configuration file for the server.</td>
</tr>
<tr>
<td><code>-d</code></td>
<td>Enables debugging.</td>
</tr>
<tr>
<td><code>-e rli_expire_int</code></td>
<td>Interval (seconds) at which an RLI server should expire stale entries.</td>
</tr>
<tr>
<td><code>-F lrc_update_factor</code></td>
<td>If <code>lrc_update_immediate</code> mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (<code>update_ll_int</code>) is multiplied by <code>lrc_update_factor</code>.</td>
</tr>
<tr>
<td><code>-f maxfreethreads</code></td>
<td>Maximum number of idle threads the server will leave running. Excess threads are terminated.</td>
</tr>
<tr>
<td>`-I true</td>
<td>false`</td>
</tr>
<tr>
<td><code>-i idletimeout</code></td>
<td>Seconds after which idle client connections are timed out.</td>
</tr>
<tr>
<td><code>-K rlskeyfile</code></td>
<td>Name of the X.509 key file. Sets environment variable <code>X509_USER_KEY</code>.</td>
</tr>
<tr>
<td><code>-L loglevel</code></td>
<td>Sets the log level. By default this is 0, which means only errors will be logged. Higher values mean more verbose logging.</td>
</tr>
<tr>
<td>`-l true</td>
<td>false`</td>
</tr>
<tr>
<td><code>-M maxconnections</code></td>
<td>Maximum number of active connections. It should be small enough to prevent the server from running out of open file descriptors.</td>
</tr>
<tr>
<td><code>-m maxthreads</code></td>
<td>Maximum number of threads server will start up to support simultaneous requests.</td>
</tr>
<tr>
<td><code>-N</code></td>
<td>Disables authentication checking.</td>
</tr>
<tr>
<td><code>-o update_buftime</code></td>
<td>LRC to RLI updates are buffered until either the buffer is full or this much time (in seconds) has elapsed since the last update.</td>
</tr>
<tr>
<td><code>-p pidfiledir</code></td>
<td>Directory where PID files should be written.</td>
</tr>
</tbody>
</table>
**-r**  Configures whether the server is an RLI server.

The default value is **false**.

**-S rli_expire_stale**  Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC).

Stale entries are not returned in queries.

**-s startthreads**  Number of threads to start up initially.

**-t timeout**  Timeout (in seconds) for calls to other RLS servers (in other words, for LRC calls to send an update to an RLI).

A value of **0** disables timeouts.

The default value is **30**.

**-U myurl**  URL for this server.

**-u update_ll_int**  Interval (in seconds) between lfn-list LRC to RLI updates.

**-v**  Shows version and exits.
Chapter 3. Configuring RLS

1. Configuration overview

RLS configuration involves statically-defined, system settings as defined in the RLS configuration file (see $GLOBUS_LOCATION/etc/globus-rls-server.conf), settings changed temporarily at run-time using the RLS Admin tool (see globus-rls-admin(1) -C option value command), and finally LRC-to-RLI and RLI-to-RLI updates configured using the RLS Admin tool (see globus-rls-admin(1) -a, -A, -d commands).

2. Server configuration file (globus-rls-server.conf)

Configuration settings for the RLS are specified in the globus-rls-server.conf file. If the configuration file is not specified on the command line (see the -c option) then it is looked for in both:

• $GLOBUS_LOCATION/etc/globus-rls-server.conf
• /usr/local/etc/globus-rls-server.conf if GLOBUS_LOCATION is not set

Note

Command line options always override items found in the configuration file.

The configuration file is a sequence of lines consisting of a keyword, whitespace, and a value. Comments begin with # and end with a newline.

3. Basic configuration

Review the server configuration file $GLOBUS_LOCATION/etc/globus-rls-server.conf and change any options you want. The server man page globus-rls-server(8) has complete details on all options. The complete details are also provided later in this section.

A minimal configuration file for both an LRC and RLI server would be:

```
# Configure the database connection info
db_user       dbuser
db_pwd        dbpassword

# If the server is an LRC server
lrc_server    true
lrc_dbname    lrc1000

# If the server is an RLI server
rli_server    true
rli_dbname    rli1000 # Not needed if updated by Bloom filters

# Configure who can make requests of the server
acl .*: all
```
# RE matching grid-mapfile users or DNs from x509 certs

...  

## 4. Host key and certificate configuration

The server uses a host certificate to identify itself to clients. By default this certificate is located in the files `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem`. Host certificates have a distinguished name of the form `/CN=host/FQDN`. If the host you plan to run the RLS server on does not have a host certificate, you must obtain one from your Certificate Authority. The RLS server must be run as the same user who owns the host certificate files (typically root). The location of the host certificate files may be specified in `$_LOCATI$ION/etc/globus-rls-server.conf`:

```
rlscertfile     path-to-cert-file   # default /etc/grid-security/hostcert.pem
rlskeyfile      path-to-key-file    # default /etc/grid-security/hostkey.pem
```

It is possible to run the RLS server without authentication, by starting it with the `-N` option, and using URL's of the form `rlsn://server` to connect to it. Notice that the URL scheme is `rlsn` as opposed to `rls`.

It is generally recommended to run the server with a user account other than root for added security. In order to do so, you will need to create complimentary key and certificate files owned by a designated user account, `globus` for instance.

1. Begin by copying the `/etc/grid-security/hostcert.pem` and `/etc/grid-security/hostkey.pem` to `/etc/grid-security/containercert.pem` and `/etc/grid-security/containerkey.pem`. Note that we use the prefix "container" to conform with the recommended naming scheme for other services distributed with the Globus Toolkit.

   ```
   % cp /etc/grid-security/hostcert.pem /etc/grid-security/containercert.pem
   % cp /etc/grid-security/hostkey.pem /etc/grid-security/containerkey.pem
   ```

2. Then change ownership of the files to the designated user account, `globus` in our example.

   ```
   % chown globus /etc/grid-security/containercert.pem
   % chown globus /etc/grid-security/containerkey.pem
   ```

3. Change the `rlskeyfile` and `rlscertfile` settings in the RLS configuration file (`$_LOCATI$ION/etc/globus-rls-server.conf`) to reflect the appropriate filenames.

   ```
   rlscertfile     /etc/grid-security/containercert.pem
   rlskeyfile      /etc/grid-security/containerkey.pem
   ```

4. Finally, bear in mind that your certificate and key files must always have file permissions 644 and 400 respectively.
If authentication is enabled, RLI servers must include acl configuration options that match the identities of LRC servers that update it and that grant the `rli_update` permission to the LRCs.

## 5. Configuring LRC to RLI updates

One of the key benefits to using the RLS for managing replica location information is its distributed architecture. In a distributed deployment, one or more Local Replica Catalog (LRC) services will send updates of its contents to one or more Replica Location Index (RLI) services.

By default the installed LRC is not configured to send updates to any RLI, even the local RLI co-located with the local LRC. Use the `globus-rls-admin(1)` tool to configure the LRC to send updates to one or more RLI services.

- **To configure the LRC to send uncompressed lists of its logical names to a RLI**, use the following command:

  ```bash
  % $GLOBUS_LOCATION/sbin/globus-rls-admin -a rls://rli_host rls://lrc_host
  ```

- **To configure the LRC to send compressed bitmaps (using Bloom filters) of its logical names to a RLI**, use the following command:

  ```bash
  % $GLOBUS_LOCATION/sbin/globus-rls-admin -A rls://rli_host rls://lrc_host
  ```

- **To configure the LRC to stop sending updates to a RLI**, use the following command:

  ```bash
  % $GLOBUS_LOCATION/sbin/globus-rls-admin -d rls://rli_host rls://lrc_host
  ```

### Note

While any given LRC is capable of sending uncompressed or compressed updates to any RLI. The RLI service must be configured to accept either uncompressed or compressed updates but not both. See the `rli_bloom-filter` setting of the RLS configuration file for more details.

There are tradeoffs between using uncompressed and compressed updates in your configuration. The advantage of using compressed updates, not surprisingly, is a significant reduction in network overhead and memory usage. As replica location mappings grow into the 10's of millions or more, the savings of using compressed updates becomes important. On the other hand, due to the compressed nature of the Bloom filter bitmap used to represent the logical names in the LRC, the wildcard query at the RLI cannot be supported when update compression is used.
6. Configuring the RLS Server for the MDS2 GRIS

The server package includes a program called `globus-rls-reporter` that will report information about an RLS server to the MDS2 GRIS. Use this procedure to enable this program:

1. To enable Index Service reporting, add the contents of the file `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf` to the MDS2 GRIS configuration file `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf`.

2. If necessary, set your virtual organization (VO) name in `$GLOBUS_LOCATION/setup/globus/rls-ldif.conf`. The default value is `local`. The VO name is referenced twice, on the lines beginning `dn:` and `args:`.

3. You must restart your MDS (GRIS) server after modifying `$GLOBUS_LOCATION/etc/grid-info-resource-ldif.conf` You can use the following commands to do so:

   ```
   $GLOBUS_LOCATION/sbin/SXXgris stop
   $GLOBUS_LOCATION/sbin/SXXgris start
   ```

7. Complete RLS Server settings (`globus-rls-server.conf`)

This section describes the complete details of the RLS Server configuration settings.
Table 3.1. Complete RLS Server settings (globus-rls-server.conf)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| acl user: permission [permission] | acl entries may be a combination of DNs and local usernames. If a DN is not found in the gridmap file then the file is used to search the acl list. A gridmap file may also be used to map DNs to local usernames, which in turn are matched against the regular expressions in the acl list to determine the user's permissions. user is a regular expression matching distinguished names (or local usernames if a gridmap file is used) of users allowed to make calls to the server. There may be multiple acl entries, with the first match found used to determine a user's privileges. [permission] is one or more of the following values:  
  - lrc_read Allows client to read an LRC.  
  - lrc_update Allows client to update an LRC.  
  - rli_read Allows client to read an RLI.  
  - rli_update Allows client to update an RLI.  
  - admin Allows client to update an LRC's list of RLIs to send updates to.  
  - stats Allows client to read performance statistics.  
  - all Allows client to do all of the above. |
| authentication true|false | Enable or disable GSI authentication. The default value is true.  
If authentication is enabled (true), clients should use the URL schema rls: to connect to the server.  
If authentication is not enabled (false), clients should use the URL schema rlsn:. |
| db_pwd password | Password to use to connect to the database server. The default value is changethis. |
| db_user databaseuser | Username to use to connect to database server. The default value is dbperson. |
| idletimeout seconds | Seconds after which idle connections close. The default value is 900. |
| loglevel N | Sets loglevel to N (default is 0). Higher levels mean more verbosity. |
| **logtype syslog|syslog-ng** | Sets system log type. (default is syslog).

syslog configures RLS to use the syslog facility. This is the default.

syslog-ng configures RLS to use the syslog-ng facility. |
| **lrc_bloomfilter_num-hash N** | Number of hash functions to use in Bloom filters.

The default value is 3.

Possible values are 1 through 8.

This value, in conjunction with lrc_bloomfilter_ratio, will determine the number of false positives that may be expected when querying an RLI that is updated via Bloom filters.

*Note:* The default values of 3 and 10 give a false positive rate of approximately 1%.

| **lrc_bloomfilter_ratio N** | Sets ratio of bloom filter size (in bits) to number of LFNs in the LRC catalog (in other words, size of the Bloom filter as a multiple of the number of LFNs in the LRC database.) This is only meaningful if Bloom filters are used to update an RLI. Too small a value will generate too many false positives, while too large a value wastes memory and network bandwidth.

The default value is 10.

*Note:* The default values of 3 and 10 give a false positive rate of approximately 1%.

| **lrc dbname** | Name of LRC database.

The default value is lrcdb. |
| **lrc_server true|false** | If LRC server, the value should be true.

The default value is false. |
| **lrc_update_factor N** | If lrc_update_immediate mode is on, and the LRC server is in sync with an RLI server (an LRC and RLI are synced if there have been no failed updates since the last full soft state update), then the interval between RLI updates for this server (update_ll_int) is multiplied by the value of this option. |
| **lrc_update_immediate true|false** | Turns LRC to RLI immediate mode updates on (true) or off (false).

The default value is false. |
| **lrc_update_retry seconds** | Seconds to wait before an LRC server will retry to connect to an RLI server that it needs to update.

The default value is 300. |
| **maxbackoff seconds** | Maximum seconds to wait before re-trying listen in the event of an I/O error.

The default value is 300. |
| **maxfreethreads N** | Maximum number of idle threads. Excess threads are killed.

The default value is 5. |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>maxconnections N</td>
<td>Maximum number of simultaneous connections.</td>
<td>100</td>
</tr>
<tr>
<td>maxthreads N</td>
<td>Maximum number of threads running at one time.</td>
<td>30</td>
</tr>
<tr>
<td>myurl URL</td>
<td>URL of server.</td>
<td>rls://&lt;hostname&gt;:port</td>
</tr>
<tr>
<td>odbcini filename</td>
<td>Sets environment variable ODBCINI.</td>
<td>If not specified, and ODBCINI is not already set, then the default value is $GLOBUS_LOCATION/var/odbc.ini.</td>
</tr>
<tr>
<td>pidfile filename</td>
<td>Filename where pid file should be written.</td>
<td>$GLOBUS_LOCATION/var/&lt;programname&gt;.pid</td>
</tr>
<tr>
<td>port N</td>
<td>Port the server listens on.</td>
<td>39281</td>
</tr>
<tr>
<td>result_limit limit</td>
<td>Sets the maximum number of results returned by a query.</td>
<td>0 (zero), which means no limit.</td>
</tr>
<tr>
<td></td>
<td>If a query request includes a limit greater than this value, an error (GLOBUS_RLS_BADARG) is returned.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If the query request has no limit specified, then at most result_limit records are returned by a query.</td>
<td></td>
</tr>
<tr>
<td>rli_bloomfilter</td>
<td>RLI servers must have this set to accept Bloom filter updates.</td>
<td>If true, then only Bloom filter updates are accepted from LRCs.</td>
</tr>
<tr>
<td>true/false</td>
<td>If false, full LFN lists are accepted.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><em>Note: If Bloom filters are enabled, then the RLI does not support wildcarded queries.</em></td>
<td></td>
</tr>
</tbody>
</table>
| **rli_bloomfilter_dir** | If an RLI is configured to accept bloom filters (`rli_bloomfilter true`), then Bloom filters may be saved to this directory after updates.  

This directory is scanned when an RLI server starts up and is used to initialize Bloom filters for each LRC that updated the RLI.  

This option is useful when you want the RLI to recover its data immediately after a restart rather than wait for LRCs to send another update.  

If the LRCs are updating frequently, this option is unnecessary and may be wasteful in that each Bloom filter is written to disk after each update.  

- **none**  
  Bloom filters are not saved to disk.  
  This is the default.  
- **default**  
  Bloom filters are saved to the default directory:  
  
  - `$GLOBUS_LOCATION/var/rls-bloomfilters` if `GLOBUS_LOCATION` is set  
  - else, `/tmp/rls-bloomfilters`  
- **pathname**  
  Bloom filters are saved to the named directory.  
  Any other string is used as the directory name unchanged.  
  The Bloom filter files in this directory have the name of the URL of the LRC that sent the Bloom filter, with slashes(`/`) changed to percent signs(`%`) and ".bf" appended. |

| **rli_dbname database** | Name of the RLI database.  

The default value is `rlidb`. |

| **rli_expire_int seconds** | Interval (in seconds) between RLI expirations of stale entries. In other words, how often an RLI server will check for stale entries in its database.  

The default value is `28800`. |

| **rli_expire_stale seconds** | Interval (in seconds) after which entries in the RLI database are considered stale (presumably because they were deleted in the LRC).  

The default value is `86400`.  

This value should be no smaller than `update_ll_int`.  

Stale RLI entries are not returned in queries.  

*Note:* If the LRC server is responding, this value is not used. Instead the value of `update_ll_int` or `update_bf_int` is retrieved from the LRC server, multiplied by 1.2, and used as the value for this option. |
### Configuring RLS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rli_server</td>
<td>If an RLI server, the value should be <code>true</code>. The default value is <code>false</code>.</td>
</tr>
<tr>
<td>rlscertfile file-name</td>
<td>Name of the X.509 certificate file identifying the server. This value is set by setting environment variable <code>X509_USER_CERT</code>.</td>
</tr>
<tr>
<td>rlskeyfile filename</td>
<td>Name of the X.509 key file for the server. This value is set by setting environment variable <code>X509_USER_KEY</code>.</td>
</tr>
<tr>
<td>startthreads N</td>
<td>Number of threads to start initially. The default value is 3.</td>
</tr>
<tr>
<td>timeout seconds</td>
<td>Timeout (in seconds) for calls to other RLS servers (e.g., for LRC calls to send an update to an RLI).</td>
</tr>
<tr>
<td>update_bf_int seconds</td>
<td>Interval in seconds between LRC to RLI updates when the RLI is updated by Bloom filters. In other words, how often an LRC server does a Bloom filter soft state update. This can be much smaller than the interval between updates without using Bloom filters (<code>update_ll_int</code>). The default value is 300.</td>
</tr>
<tr>
<td>update_buftime seconds</td>
<td>LRC to RLI updates are buffered until either the buffer is full or this much time in seconds has elapsed since the last update. The default value is 30.</td>
</tr>
<tr>
<td>update_ll_int seconds</td>
<td>Number of seconds before an LRC server does an LFN list soft state update. The default value is 86400.</td>
</tr>
</tbody>
</table>
### Table A.1. Replica Locator Service (RLS) Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error with credential: The proxy credential: &lt;credential&gt; with subject: &lt;subject&gt; expired &lt;minutes&gt; minutes ago</td>
<td>Expired proxy credential</td>
</tr>
<tr>
<td>Unable to connect to local host:xxxx</td>
<td>Unable to connect to the local host. This can be due to a variety of reasons, including a wrong address or port number in the RLS connection URL or an issue with a firewall configuration.</td>
</tr>
</tbody>
</table>
| "connection timeout" | At times, a client may experience a connection timeout when interacting with the RLS server due to a variety of reasons:  
  - One reason could simply be due to wide-area network latency or congestion.  
  - Another situation that users eventually encounter is due to scaling of the system. As the RLS server's database of replica location mappings grows in size, some query operations, such as bulk queries involving large quantities of mappings or wildcard queries that result in a large subset of mappings, will begin to take more time both to process the query and to return the large results set to the client over the network. |

<table>
<thead>
<tr>
<th>Possible Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new proxy with grid-proxy-init.</td>
</tr>
</tbody>
</table>
| • Double-check the address and port number in the RLS connection URL, parameters are correct.  
  • If a firewall configuration is preventing connections to the target host for a particular port, you may need to consult the system administrator. |
| If timeouts are experienced with increasing frequency, increase the RLS server's timeout configuration parameter found in the $GLOBUS_LOCATION/var/globus-rls-server.conf file. You may also use the \texttt{-t} timeout option of the \texttt{globus-rls-cli} tool. |
Glossary

B

Bloom filter  Compression scheme used by the Replica Location Service (RLS) that is intended to reduce the size of soft state updates between Local Replica Catalogs (LRCs) and Replica Location Index (RLI) servers. A Bloom filter is a bit map that summarizes the contents of a Local Replica Catalog (LRC). An LRC constructs the bit map by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits.

L

Local Replica Catalog (LRC)  Stores mappings between logical names for data items and the target names (often the physical locations) of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. Also may associate attributes with logical or target names. Each LRC periodically sends information about its logical name mappings to one or more RLIs.

See also RLI3.

logical file name  A unique identifier for the contents of a file.

P

physical file name  The address or the location of a copy of a file on a storage system.

R

Replica Location Index (RLI)  Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents.

RLS attribute  Descriptive information that may be associated with a logical or target name mapping registered in a Local Replica Catalog (LRC). Clients can query the LRC to discover logical names or target names that have specified RLS attributes.

3 #ii