DataGRID

USER GUIDE FOR EDG REPLICA MANAGER 1.8.1

WP2 EDG REPLICA MANAGER

Document identifier:	**DataGrid-02-ERM-USER-GUIDE**
EDMS id:	
Date:	February 4, 2005
Work package:	**WP2: Data Management**
Partner(s):	**CERN, PPARC, HIP, INFN, ITC, KDC**
Lead Partner:	**CERN**
Document status:	**FINAL**
Author(s):	WP2
File:	**edg-replica-manager-userguide**

Abstract: This document gives an overview of how to use the EDG Replica Manager.

CONTENTS

CONTENTS

Document Change Record

Issue	Date	Comment	Author
0_1	4/03/03	Expanded with this DCR, Glossary and reworked the Intro. Previous version by Kurt Stockinger.	Peter Kunszt
0_2	25/03/03	Section on EDG Replica Manager vs Reptor added, some small fixes, new option -vo added	Heinz & Kurt Stockinger
1_0	26/03/03	Options –vo and –config described	Peter Kunszt
1_0_1	28/03/03	Description of the log file usage added	Peter Kunszt
1_0_2	28/04/03	Description of few new commands added	Kurt Stockinger
1_0_3	5/06/03	Minor editorial changes	Heinz Stockinger
1_0_4	12/06/03	Examples for file names added already in the beginning of the document. Several editorial changes, additions and corrections of commands.	Heinz Stockinger
1_1_0	16/06/03	Changes for file-naming - we now use SURL instead of SFN with the following syntax srm://hostname/path/filename	Heinz Stockinger
1_2_0	25/06/03	Documentation for the command “list” added	Heinz Stockinger
1_3_0	18/08/03	Documentation for the commands “getTurl” and bulkCopyAndRegisterFile added; additions to “getBestFile”	Heinz Stockinger
1_4_0	18/09/03	Add documentation for Test Suite; adapt change in deleteFile interface	Heinz Stockinger
1_4_1	19/09/03	reflect changes in CLI test suite	Heinz Stockinger
1_4_1	19/09/03	add logging switches, update configuration file location; update VOMS cert info; remove box indicating autoloading of conf file in current dir (disabled); add validate option; it’s the next release - remove deprecated commands	Peter Kunszt
1_5_0	09/10/03	changes and tidy up for 2.1 release	David Cameron

1. INTRODUCTION

The data management services of WP2 are a set of coordinated high-level grid middleware services for Grid data management, including file I/O, mass storage, data access and replication. These services are:

- Replica Location Service (RLS)
- Replica Metadata Catalog (RMC)
- Replica Optimization Service (ROS)

The framework coordinating file cataloguing, replication and copying is called the EDG Replica Manager (or Reptor) [1]. Reptor is the Grid user's entry point to the entire set of WP2 services. In this document the functionality and usage of the Reptor is described in detail. For details of the Reptor service design, we refer the reader to [1, 2, 7].

1.1. DOCUMENT STRUCTURE

This document is structured in the following way:

- The **Glossary** Section 1.2. gives background information on terminology and file naming. You will also find some examples there. The section is rather useful for beginners.
- In a similar way, also Section 1.3. provides useful background information on file cataloging.
- Section 2. (**User Interface**) is very important if you are using the tool for the first time since it gives all basic information of how the command line interface works. In addition, a simple **Quick Start Guide** and **How To** is provided in Section 4. We recommend that you read both of these sections carefully.

  Details for each of the commands (with all possible arguments, options etc.) provided by the EDG Replica manager are given in Section 3. **Commands**.
- For people already familiar with the previous Replica Manager provided by WP2 in EDG release 1.4.x (called `edg-replica-manager`), some hints are given in the Appendix in Section 6. in order to provide an easier transition to this new tool described in this user guide.

1.2. GLOSSARY

1.2.1. DATA MANAGEMENT SERVICES

SRM	**Storage Resource Manager**	A high-level interface to a storage system.
RLS	**Replica Location Service**	The distributed service providing the mappings between GUIDs and SURLs. An RLS has two components: Local Replica Catalogs and Replica Location Indexes.
LRC	**Local Replica Catalog**	The catalog storing GUID to SURL mappings, along with SURL attributes for a given site, or a single Storage Resource Manager at a site. It only stores GUID to SURL mappings for SURLs that are actually located in the given site or SRM.
RLI	**Replica Location Index**	The catalog storing information about which Local Replica Catalogs have GUID to SURL mappings for a particular GUID. It thus provides the link between different LRCs, allowing for distributed indexing and querying of the Catalogs.
RMC	**Replica Metadata Catalog**	The catalog storing LFN aliases for GUID, as well as attributes on GUIDs and LFNs.
ROS	**Replica Optimization Service**	A service providing information to guide selection between replicas located at different sites. This is based on network information collected from available network monitors.

1.2.2. DATA NAMING

SURL	**Storage URL**	An SURL is a locator for a physical file, where the scheme specific part is understood by a Storage Resource Manager (SRM). It is a URL where the scheme is ‘srm’ and the host is a valid SRM host.
UUID	**Universally Unique IDentifier**	A UUID is a 128 bits long number, and is either guaranteed to be different from all other UUIDs generated until 3400 A.D. or extremely likely to be different (depending on the mechanism chosen to generate it).
GUID	**Grid Unique IDentifier**	A UUID generated by the Replica Management System for an SURL. It is created at the SURL registration time. A GUID is immutable.
LFN	**Logical File Name**	A Logical File Name is a user defined alias to a GUID. Unlike GUIDs, aliases are mutable but they still should be globally unique. Since the Replica Management System has no control over the creation of LFNs, this global uniqueness is only weakly enforced.
TURL	**Transport URL**	A Transport URL is returned by a SRM in response to a request for a way to access a SURL. It includes the actual protocol you can access the SURL by. For instance, ‘gsiftp’ for GridFTP, or ‘http’ for HTTP access.

EXAMPLES

In the following we give examples for the definitions of naming conventions.

SURL examples:

```
srm://host1.cern.ch/directory1/directory2/filename
srm://lxhare0384.cern.ch/flatfiles/cms/data/05/x.dat
```

GUID examples:

```
guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a
guid:7c29f32b-4964-11d7-a86c-9ee9a33b1f19
```

LFN examples:

```
lfn:mydata
lfn://any_name_you_want
```

TURL examples:

```
gsiftp://host1.cern.ch/directory1/directory2/filename
http://lxhare0384.cern.ch/flatfiles/cms/data/05/x.dat
```

1.3. FILE CATALOGING

The EDG Replica Manager manages data held in files. In a Grid environment the data files are replicated, possibly on a temporary basis, to many different sites depending on where the data is needed. The users or applications do not need to know where the data is located. Having a logical name for the data file they should be able to use the data management services to locate and access the data.

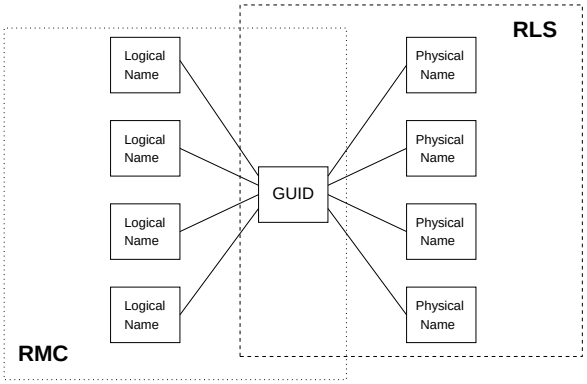

Figure 1: The Logical File Name to GUID mapping is maintained in the Replica Metadata Catalog, the GUID to Physical File Name mapping in the RLS.

Users and applications can use the Grid Unique IDentifier (GUID) to locate the data; these are assigned at data registration time and are based on the UUID standard to guarantee unique IDs. GUIDs are immutable. All the replicas of the data will share the same GUID.

In order to locate a Grid accessible file, the human user will normally use a Logical File Name (LFN). LFNs are usually more intuitive, human-readable strings as opposed to the cryptic GUIDs, and are allocated by the user. Within the Grid LFNs must be globally unique as well, each LFN being an alias for exactly one GUID. The logical name space can be partitioned at will by the users. A GUID may have many LFNs as aliases. In this document we also call LFNs 'GUID aliases'.

Given either a LFN or GUID, the Metadata and Replica Location Catalogs can be contacted to obtain a Storage URL (SURL) for the physical instance(s) of the required file. The SURL is a URI having the Storage Resource Manager as its hostname which takes responsibility for the storage of the physical file. This Storage Resource Manager (SRM) should be able to accept the SURL and return a valid URL, a Transport URL (TURL) in our terminology, to the application. This TURL provides all the information required (i.e. protocol, host, port and path) so that the application can open and retrieve the required physical file.

Note that Reptor requires both catalogs, Replica Location Service and Replica Metadata Catalog, but an end-user should not need to distinguish between the two of them. In the remainder of the document we use the terms **Replica Catalog** or **Catalog** to refer to both of them regardless of their exact functionality.

2. USER INTERFACE

We offer command line interfaces (CLIs) and application program interfaces (APIs) to the EDG Replica Manager. In this user-guide we only discuss the CLIs and their functionality, the APIs (that are very similar) are described in the developer's guide.

2.1. DIFFERENCES TO THE PREVIOUS EDG REPLICA MANAGER

The command line executable, `edg-replica-manager` or `edg-rm` is by design very similar to its predecessor (edg-replica-manager [5]) with the same name on the previous release of the Testbed. The new EDG Replica Manager has more functionality and the arguments are usually parsed by placement, not by options as before. Most of the command names have been preserved but the arguments have changed so read this guide carefully if you are migrating your jobs and scripts. The detailed differences are in the appendix.

2.2. COMMAND LINE INTERFACE

The command line interface to the EDG Replica Manager can be accessed via the `edg-replica-manager` command, or `edg-rm` for short. It takes a set of global options and then a command that has in turn options of its own.

The global options are listed if the command is given without any arguments:

```
usage: edg-replica-manager [options] command [options]
    --log-debug       enable debug-level logging
    --log-info        enable info-level logging
    --log-off         disable logging
 -h,--help            print help (if command is given, details on command)
 -i,--insecure        Connect in an insecure manner, i.e. not https.
    --config=<file>   read configuration from specified file
    --vo=<VO>         set Virtual Organization
 -v,--verbose         print additional information while executing
```

> Note that currently the VO needs to be passed to the replica manager, e.g. --vo=cms. The vo option will not be necessary once VOMS is available. Then the VOMS certificate is parsed and the VO is extracted. If the --vo option is still given, it is ignored and the VOMS VO is the one which is chosen.

The default configuration file will be read from `$EDG_LOCATION/var/etc/edg-replica-manager/edg-replica-manager.conf` where $EDG_LOCATION is the installation location of Reptor (/opt/edg for a default installation).

The following commands (long or short forms) can be used in Reptor (the same description is printed using the -h option):

Management commands		
copyAndRegisterFile	cr	Put a local file into Grid Storage and register it in the Catalog
bulkCopyAndRegister	bcr	Do multiple copyAndRegisters.
replicateFile	rep	Replicate an existing file to a certain Grid Storage, update Catalog
deleteFile	del	Delete a file from Storage and remove entry from Catalog
getTurl	gt	Get a TURL given a SURL and a protocol.
Catalog commands		
registerFile	rf	Register a file in the catalog
registerGUID	rg	Register a file with a known GUID in the catalog
unregisterFile	uf	Unregister a file from the catalog
listReplicas	lr	List all replicas of a logical file name
listGUID	lg	List the GUID of a known LFN or SURL
addAlias	add	Add an LFN alias to an existing GUID
removeAlias	ra	Remove an alias to GUID mapping
printInfo	pi	Print all info service data to screen or to a file that can be reused by the static configuration
getVersion		Get the replica manager version
validate		Validate for correctness and normalize file names
Optimization commands		
getBestFile	gbf	Replicate a file to the ‘nearest’ Storage Element in the ‘cheapest’ way
listBestFile	lbf	List the replica that has the smallest access cost
getAccessCost	ac	List access costs for all replicas
File transfer commands		
copyFile	cp	Copy a file. Use this to copy a file to local store by specifying an LFN as the source and a local destination.
list	ls	List the contents of a directory on an SE (SRM or GridFTP)

In the following sections these commands are described in detail.

3. COMMANDS

3.1. MANAGEMENT COMMANDS

The management commands are:

- `copyAndRegisterFile`
- `replicateFile`
- `deleteFile`
- `getTurl`

3.1.1. `copyAndRegisterFile`

Command:	**copyAndRegisterFile**
Description:	Put a (local) file into Grid Storage and register it in the Catalog
Arguments:	**sourceFileName** The file to copy and register. It should be a fully qualified URI. Possible schemes are: file, gsiftp, http, https, ftp.
Options:	**-d** or **--destination** *destination-SURL* or *SE host* **-l** or **--logical-file-name** *logical-file-name* **-p** or **--protocol** *protocol* **-n** or **--streams** *number of streams*
Example:	`edg-replica-manager --insecure --vo=wpsix copyAndRegisterFile file:/home/bob/analysis/data5.dat -d lxshare0384.cern.ch`

COMMAND LINE TOOL

```
usage: edg-replica-manager copyAndRegisterFile sourceFileName [options]
Copy a file to grid-aware storage and register it in the grid catalog. Upon
successful completion this call returns the GUID of this file that can be
used to retrieve it again.
 -d,--destination-file <file>     The destination file name (SURL) or
                                  destination SE host. If it is not given,
                                  the closeSE is found from the info
                                  services and an automatic file name is
                                  used as an SURL.
 -l,--logical-file-name <file>    The logical file name to register this
                                  file with. Has to start with 'lfn:'. If it
                                  is not given, the only way to find a file
                                  is through the GUID that is returned by
                                  this call.
 -n,--streams <#streams>          The number of parallel streams to use.
                                  Defaults to 8. This only has an effect if
                                  the protocol supports parallel streams.
                                  Currently the only protocol to do so is
                                  gsiftp.
 -p,--protocol <protocol>         The protocol to be used for transfer.
                                  Defaults to gsiftp.
```

```
Arguments:
  sourceFileName        The file to copy and register. It should be a fully
                        qualified URI. Possible schemes are: file, gsiftp,
                        http, https, ftp.
```

The `copyAndRegisterFile` call performs the task of copying a file into grid-aware storage and registering the copy in the Replica Catalog as an atomic operation. This is the preferred method to 'bring files into the grid'.

The same could be done by issuing `copyFile` and `registerFile` in sequence, but since `copyFile` may not have an SE or SURL as its destination, the user would need to specify a valid transport file name (using gridftp for example) to a location that happens to be the proper location for the file and then register it using `registerFile`. That is much more error prone and cumbersome than this method.

As a rule, `copyFile` should be used to copy files around to non-grid aware storage, or to simply copy files *out* of the grid (i.e. the source may be grid-aware but not the destination).

`registerFile` should be used to register files that are already in the grid-aware store before it was made grid-aware or to register files that appear there through other means (like new data files). If the files that need to be put into the grid are not yet in their grid-aware location, `copyAndRegisterFile` is the most robust method to use.

Semantics

`copyAndRegisterFile` also handles all the possible failures if the registration should fail after the copy has taken place (i.e. the copied file is registered using another name or if that is not possible it is actually removed again). If the destination already exists, the operation fails.

Re-registrations of existing data files are possible by issuing `copyFile` and `registerFile` in sequence as specified above, or by copying the file 'out' using `copyFile` and re-registering it again using this method but storing it with a different name. If users really want to do a re-registration resulting in having more than one GUID for the same data (of course not for the same file), it can be done this way but we knowingly make this operation difficult so that people don't have more than one GUID referring to the same data by mistake.

Since only files on Storage Elements may be registered, `--destination-file` must refer to a Storage Element just like for `replicateFile`. The difference to `replicateFile` is that the latter requires the `sourceFileName` to be a valid grid file descriptor, i.e. either an LFN, GUID or other SURL, while `copyAndRegisterFile` does explicitly forbid such sources and accepts only non-grid files. For files already in the grid, `replicateFile` must be used.

The call returns the GUID of the new entry.

Arguments

`sourceFileName` The file to be registered. It must be accessible through this name using the specified protocol in the URI. The `sourceFileName` may be:

- A local file (i.e. a URI with `file` as its schema).
- A transport URI with a valid protocol schema (http, ftp, etc.).

Options

`destination-file` The physical destination file. `destination-file` may be specified in three different ways:

- As a fully qualified Storage URL (SURL). If this file already exists, the operation fails.

- Only the host of the SE where the file should be stored, i.e. the URI was constructed just by giving it a host string, which will be stored in the URI path. The storage location will be determined automatically depending on the user's VO and a suitable file name will be chosen.
- If no destination is specified at all (i.e. the option is omitted) the 'closest' SE is located and the file is stored as in the previous case.

`logical-file-name` The logical (file) name to be used in the catalog to find the file again. If this option is omitted, only the GUID returned by this call will be available to retrieve the file.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure. It also prints the GUID that this file was registered with on stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `sourceFileName` does not exist.
- The `sourceFileName` or `destination-file` are invalid URIs.
- The `sourceFileName` or `destination-file` cannot be accessed.
- The copy operation fails.
- The given `logicalFileName` already exists.

3.1.2. `bulkCopyAndRegisterFile`

Command:	**bulkCopyAndRegisterFile**
Description:	Put many files into Grid Storage and register them in the Catalog, all in one atomic operation.
Arguments:	**filelist** A file containing a list of local files to be bulk uploaded. This should be specified as a URI (with scheme 'file').
Options:	**-d** or **--destination** *destination-SURL* or *SE host* **-s** or **--logical-filename-suffix** *suffix* **-l** or **--logical-file-names** *filename* **-p** or **--protocol** *protocol* **-n** or **--streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix bulkCopyAndRegisterFile file:///home/mydir/myfile-list`

COMMAND LINE TOOL

```
usage: edg-replica-manager bulkCopyAndRegister filelist [command-options]
Bulk upload multiple files to a given SE and register them in the replica
catalog.
 -d,--destination-host <host>            The destination SE host. If it is
                                         not given, the closeSE is found
                                         from the info services. The
                                         destination SURLs are always
                                         automatically generated.
 -s,--logical-filename-suffix <suffix>   The logical filename suffix to
                                         register this entry with. The LFN
                                         will be constructed using the full
                                         filename plus this suffix. If this
                                         option is not specified, no LFN
                                         will be registered. This cannot be
                                         used with the --logical-files (-l)
                                         option.
 -l,--logical-files <filename>           A file containing the list of LFNs
                                         to register for each source file.
                                         This must have the same number of
                                         entries as filelist. This filename
                                         should be specified as a URI (with
                                         scheme 'file'). This option cannot
                                         be used with the
                                         --logical-file-name-suffix (-s)
                                         option.
 -n,--streams <#streams>                 The number of parallel streams to
                                         use. Defaults to 8. This only has
                                         an effect if the protocol supports
                                         parallel streams. Currently the
                                         only protocol to do so is gsiftp.
 -p,--protocol <protocol>                The protocol to be used for
                                         transfer. Defaults to gsiftp.
 Arguments:
  filelist              A file containing a list of local files to be bulk
                        uploaded. This should be specified as a URI (with
                        scheme 'file').
```

The functionality is very similar to the command `copyAndRegisterFile` and thus we only outline the most important differences here.

Semantics

If several files need to be "uploaded" to the Grid in one operation, running the command line `copyAndRegisterFile` might take a rather long time as each time the Java virtual machine has to be started. In order to reduce the start-up latency, the following command takes several files in one operation and thus the latency is decreased. The major difference to the command `copyAndRegisterFile` is that for possible LFNs one can only specify a suffix for all of the files rather than individual LFNs.

Arguments

A single file (identified by a URI with the scheme file) needs to be specified that contains all files that are a locally available and are to be transferred to the Grid.

Options

There two ways in which one can specify the LFNs for the set of files to be bulk uploaded:

1. The option `-s` or `--logical-filename-suffix` takes a suffix for all LFNs that are then automatically generated by suffixing the filenames given in the file list.
2. The option `-l` or `--logical-files` takes a file which contains a list of LFNs and these LFNs are used for each file uploaded to the Grid.

Return value

The call returns 0 on success and -1 on failure.

3.1.3. `replicateFile`

Command:	**replicateFile**
Description:	Replicate a file to another SE.
Arguments:	**sourceFileName** The file to replicate. This may be an LFN, GUID or SURL. For LFN and GUID the best SURL is found through listBestFile. If the destination SE is omitted, the file is replicated to the local SE, i.e. it's the same as getBestFile.
Options:	**-d** or **--destination** *destination-SURL* or *SE host* **-p** or **--protocol** *protocol* **-n** or **--streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix replicateFile lfn:mydata -d lxshare0384.cern.ch`

COMMAND LINE TOOL

```
usage: edg-replica-manager replicateFile sourceFileName [options]
Replicate a file to another SE.
 -d,--destination <file>    The destination to replicate to. This may be a
                            fully qualified SURL, just an SE host name or
                            just be omitted - in which case the replication
                            will be done to the closest SE.
 -n,--streams <#streams>    The number of parallel streams to use if
                            supported by protocol. Optional.
 -p,--protocol <protocol>   The protocol to be used for the transfer
                            (optional).
 Arguments:
  sourceFileName        The file to replicate. This may be an LFN, GUID or
                        SURL. For LFN and GUID the best SURL is found through
                        listBestFile. If the destionation SE is omitted,
                        the file is replicated to the local SE, i.e. it's
                        the same as getBestFile.
```

The `replicateFile` call performs the task of replicating a file between grid-aware stores and registering the replica in the Replica Catalog as an atomic operation. This is the preferred method to 'copy files around in the grid'.

The difference to `copyAndRegisterFile` is that this operation only allows for GUID, LFN or SURL as the source file whereas `copyAndRegisterFile` explicitly forbids that. No new GUID is generated for the replica (hence the term).

Semantics

`replicateFile` also handles all the possible failures if the registration should fail after the copy has taken place. If the remote site already contains a replica, this operation returns successfully very quickly.

Since replication can only be done between Storage Elements, both `sourceFileName` and `destination` must refer to a Storage Element (unless one uses LFN or GUID as the argument for the `sourceFileName` without further options).

The method returns the SURL of the new replica upon success or an appropriate error message upon failure.

Arguments

`sourceFileName` The file to be registered. The `sourceFileName` may be:

- A GUID. The SE will be chosen automatically.
- An LFN. The SE will be chosen automatically.
- An SURL. The given SE will be used as the source for this file.

Options

`destination-file` The physical destination may be specified in three different ways:

- As a fully qualified Storage URL (SURL). If this file already exists, the operation fails.
- The SE hostname where the file should be stored. The storage location will be determined automatically depending on the user's VO and a suitable file name will be chosen.
- If no destination is specified at all (i.e. the option is omitted) the 'closest' SE is located and the SURL is determined as in the previous case.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure. It also prints the SURL of the new replica.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `sourceFileName` does not exist.
- The `sourceFileName` or `destination-file` are invalid URIs.
- The `sourceFileName` or `destination-file` cannot be accessed.
- The copy operation fails.

3.1.4. deleteFile

Command:	**deleteFile**
Description:	Delete a file from the Grid (and thus its Catalog). All replicas of a file may be deleted by specifying the GUID and setting the --all-available flag.
Arguments:	**fileName** File to delete. This has to be a grid-file, i.e. LFN, SURL or GUID. For GUIDs the --all-available flag may be set to delete all instances. For LFNs the storage option has to be set.
Options:	**-s** or **--storage** *SE host* **-a** or **--all-available**
Example:	`edg-replica-manager --vo=wpsix deleteFile guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a --all-available`

COMMAND LINE TOOL

```
usage: edg-replica-manager deleteFile fileName [options]
Delete a file from the Grid. All replicas of a file may be deleted by
specifying the GUID and setting the --all-available flag.
 -s,--storage <host>   The storage host from which the physical instance is
                       to be deleted. This is mandatory for LFNs, ignored
                       for SURLs and mandatory for GUIDs if the --all-available
                       flag is not set.
 -a,--all-available    If this is set and the file-name is a GUID, make a
                       best-effort attempt to delete all replicas of this
                       file.
 Arguments:
  fileName              File to delete. This has to be a grid-file, i.e.
                        LFN, SURL or GUID. For GUIDs the --all-available
                        flag may be set to attempt to delete all instances
                        (on a best-effort basis). For LFNs the storage
                        option has to be set.
```

The `deleteFile` call performs the task of removing a file from grid-aware storage and unregistering the entry in the Replica Catalog as an atomic operation.

Semantics

The `fileName` argument may be either a SURL, a LFN or a GUID. If it is a SURL, only the specified file will be deleted and the corresponding GUID-SURL mapping removed from the catalog. If the `fileName` is an LFN, the `--storage` option is mandatory in order to specify which replica must be removed. The same is true if the `fileName` argument is a GUID, but for GUIDs the `--all-available` flag can be specified instead of the `storage option`, which will remove all entries in the RLS but not from the RMC! (In order to delete all LFNs from the RMC, you need to use the command removeAlias).

Arguments

`fileName` The file to be deleted. The `fileName` may be:

- A GUID. Either the `storage` or `all-available` options must be supplied.
- An LFN. The `storage` option is mandatory, the `all-available` option is not allowed.
- An SURL. The given SURL will be removed.

Options

`storage` The host name of the SE where the file resides.

`all-available` A flag specifying whether all instances of the given GUID should be removed. An error is returned if this flag is given for an LFN or SURL.

Return value

The call returns 0 on success and -1 on failure.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `fileName` does not exist.
- The `fileName` is an invalid URI.
- The `storage` host is not an SE host.
- The delete operation fails or the user has no access rights.

3.1.5. `getTurl`

Command:	**getTurl**
Description:	Get the TURL for a given SURL and transfer protocol.
Arguments:	**surl** the SURL to find the TURL for. **protocol** The protocol of the TURL.
Options:	none.
Example:	`edg-replica-manager --vo=cms getTurl srm://adc0027.cern.ch/cms/file1 gsiftp`

COMMAND LINE TOOL

```
Get the TURL for a given SURL and transfer protocol.

 Arguments:
  surl                  The SURL to find the TURL for.
  protocol              The protocol of the TURL.
```

This method returns a TURL, in order that the file with the given SURL can be accessed by the given protocol.

3.2. CATALOG COMMANDS

The catalog commands are:

- `registerFile`
- `registerGUID`

- unregisterFile
- listReplicas
- listGUID
- addAlias
- removeAlias
- printInfo
- getVersion
- validate

3.2.1. registerFile

Command:	**registerFile**
Description:	Register a file that is already on a Grid-aware store. It returns the GUID with which the file was registered. Optionally an LFN may be given as well.
Arguments:	**SURL** The SURL of the file to register.
Options:	**-l** or **--logical-file-name** *logical-file-name*
Example:	`edg-replica-manager --vo=wpsix registerFile srm://lxshare0384.cern.ch/flatfiles/alice/data/05/x.dat -l lfn:aliceprod/x`

COMMAND LINE TOOL

```
usage: edg-replica-manager registerFile SURL [options]
Register a file that already is on a Grid-aware store. It returns the GUID
with which the file was registered. Optionally an LFN may be given as well.
 -l,--logical-file-name <lfn>   The logical file name to register this file
                                with
 Arguments:
  SURL                    The SURL of the file to register.
```

Register a file in the Replica Catalog that is already stored on a Storage Element.

Semantics

The source file needs to be a qualified storage file name URI, following the rule

`srm://storage.element.host/path/file.name`

The logical name is optional; it may declare a logical identifier that can be used later to look up any instance of the file. The method returns the GUID, the Grid Unique IDentifier of the file. The details are:

- If the source file is not registered yet:
  - no logical name is given: a new GUID is generated, the file is registered and the GUID is returned to the caller.

  - a logical name is given: a new GUID will be allocated and returned, and the logical name will also be registered and can be used to retrieve the file later.

- If the file is already registered:
  - no logical name is given: the GUID that already exists is returned.
  - a logical name is given that does not exist yet: it is added as an alias in the Replica Catalog so that the file can be looked up later using that name as well.
  - a logical name is given which already exists: it is checked whether the logical name really corresponds to the file. If not, an exception is thrown, otherwise the existing GUID is returned.

This method does not include any data movement (i.e. the storage file is not copied) and assumes that the storage file is already located at a known Storage Element. If the SURL does not exist on the SE, the command fails.

By specifying an SURL that is already registered, this method can be used to add a new alias to it (in addition to the `addAlias` command)

Arguments

`SURL` The file to be registered. It must be a valid Storage URL, i.e. the host part of the given URI needs to be a recognized Storage Element, as described above.

Options

`logical-file-name` The logical alias to be included in the catalog. Using this LFN the file may be retrieved as well, in addition to its GUID. If this option is omitted, the GUID will be the only name by which the file can be retrieved.

Return value

The call returns 0 on success and -1 on failure and prints the GUID with which this file was registered to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL` does not exist.
- The `SURL or logical-file-name` is an invalid URI.
- The RLS or RMC are unattainable or ill-configured.

3.2.2. `registerGUID`

Command:	**registerGUID**
Description:	Register an SURL with a known GUID in the catalog
Arguments:	**SURL** The SURL to register. **GUID** The GUID to register the file with.
Options:	

Example:	`edg-replica-manager --vo=wpsix registerGUID`
	`srm://lxshare0384.cern.ch/flatfiles/alice/data/05/x.dat`
	`guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a`

COMMAND LINE TOOL

```
usage: edg-replica-manager registerGUID SURL GUID
Register a file that is already in a Grid-aware store with a GUID given to
the command. This is only necessary if replication could not be carried out
through replicateFile for some reason, like the data was shipped by tape.

 Arguments:
  SURL                  The SURL of the file to register.
  GUID                  The known GUID of this copy.
```

Register a file in the Replica Catalog with a known GUID. The use case for this command is the so-called *truckFTP* use case where the replication happens not over the network but through a tape delivered by FedEx (maybe this is cheaper). The remote site wants to register the received data using a known GUID and declare the data a valid replica of an existing file.

Semantics

The command takes two arguments: source file and GUID. The source file needs to be a qualified storage file name URI, following the same rule for SURLs as for `registerFile` above. The GUID needs to be known to the system already. Be careful, if the GUID corresponds to a wrong file in the remote catalog the catalog data will become inconsistent. If the SURL does not exist on the SE, the command fails.

Arguments

`SURL` The file to be registered. It must be a valid SURL as described above.

`GUID` The GUID to register the file with.

Options

None.

Return value

The call returns 0 on success and -1 on failure.

Errors

Some of the reasons for failure are:

- The user has no/expired credentials.
- The `SURL` does not exist.
- The `SURL` is an invalid URI.
- The `GUID` given is ill-formed or not a known GUID.
- The RLS or RMC are unattainable or ill-configured.

3.2.3. unregisterFile

Command:	**unregisterFile**
Description:	Unregister a file from the catalog.
Arguments:	**GUID** The GUID of the entry to unregister **SURL** The SURL to unregister
Options:	
Example:	`edg-replica-manager --vo=wpsix unregisterFile guid:1cc5353d-982f-11d7-9861-a4978670ee2b srm://lxshare0384.cern.ch/flatfiles/alice/data/05/x.dat`

COMMAND LINE TOOL

```
edg-rm -h unregisterFile:

usage: edg-replica-manager unregisterFile GUID SURL
Unregister a file from the catalog.

 Arguments:
  GUID                  The GUID to unregister
  SURL                  The matching SURL to unregister
```

Unregister a file from the Replica Location Service that is stored on a Storage Element. It takes two arguments: the GUID and SURL to which the (GUID, SURL) mapping should be removed.

Note: in order to remove/unregister an LFN, use the command `removeAlias`.

Semantics

This command removes a replica of a file from 'grid awareness'. The file stored at the SE and identified with the SURL will not be removed (this is achieved using `deleteFile`). If the SURL does not actually exist on the SE, the command will be successful anyway. If the SURL specified is the only instance (replica) of the given file, the GUID will be removed as well, otherwise only the SURL is removed from the Catalog.

Note: Since LFNs (aliases) are treated like symbolic links in the Replica Metadata Catalog, if the last GUID is deleted from the RLS, the corresponding LFNs are *not* deleted from the Replica Metadata Catalog which results in "dangling LFNs" (similar to "dangling symbolic links").

Arguments

`GUID` The GUID to unregister for a given SURL.

`SURL` The file to be unregistered. It must be a valid Storage File Name, i.e. the host part of the given URI needs to be a recognized Storage Element, as described above.

Options

None.

Return value

The call returns 0 on success and -1 on failure and prints the GUID with which this file was registered to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL or logical-file-name` is an invalid URI.
- The RLS or RMC are unattainable or ill-configured.

3.2.4. `listReplicas`

Command:	**listReplicas**
Description:	List all replicas of a file.
Arguments:	**fileName** The LFN, GUID or SURL to list all replicas of a file.
Options:	
Example:	`edg-replica-manager --vo=wpsix listReplicas lfn:mydat`

COMMAND LINE TOOL

```
usage: edg-replica-manager listReplicas fileName
List all replicas of a file.

 Arguments:
  fileName              The LFN, GUID or SURL to list all replicas of.
```

List replicas as they are registered in the Replica Catalog.

Semantics

Depending whether you specify an LFN, GUID or SURL as the input filename the edg-replica-manager will issue different commands to the underlying services. The result is always the same: a list of SURLs that are replicas of the same file.

- If a GUID is specified, then the RLS is contacted to resolve it into the corresponding list of SURLs, which is then returned and listed to stdout.
- If you specify an LFN as the filename argument, first the Replica Metadata Catalog is contacted to resolve the LFN into a proper GUID. Then the RLS is contacted to retrieve all corresponding registered SURLs.
- If an SURL is specified, the RLS is contacted to retrieve the corresponding GUID and then based on that GUID all other SURLs are found. So knowing an SURL you can find all available replicas. The SURL will be listed again in the output.

Arguments

`fileName` The filename from which to get all replicas. It must be a valid Logical File Name, GUID or Storage File Name.

Options

no options.

Return value

The call returns 0 on success and -1 on failure and prints all SURLs that are replicas of the input filename to stdout, one on each line.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL, GUID or LFN` is invalid or does not exist.
- The RLS or RMC are unattainable or ill-configured.

3.2.5. `listGUID`

Command:	**listGUID**
Description:	Print the GUID associated with an LFN or SURL.
Arguments:	**LFNorSURL** The LFN or SURL to get the GUID of.
Options:	
Example:	`edg-replica-manager --vo=wpsix listGUID lfn:mydata`

COMMAND LINE TOOL

```
usage: edg-replica-manager listGUID LFNorSURL
Get the GUID based on an LFN or SURL.

 Arguments:
  LFNorSURL              The LFN or SURL to get the GUID of.
```

Get the GUID that corresponds to a given SURL or LFN.

Semantics

Depending whether you specify an LFN or SURL as the input filename the edg-replica-manager will either access the Replica Metadata Catalog (in case of an LFN) to resolve the LFN into a GUID or the Replica Location Service (in case of an SURL) to find out the GUID with which the SURL is registered. The command always returns a GUID.

Arguments

`LFNorSURL` The LFN or Storage URL. It must be a valid URI for LFN or SURL schemes.

Options

None.

Return value

The call returns 0 on success (-1 on failure) and prints the GUID to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `SURL or LFN` is invalid or does not exist.
- The RLS or RMC are unattainable or ill-configured.

3.2.6. addAlias

Command:	**addAlias**
Description:	Add a new alias to GUID mapping
Arguments:	**GUID** The GUID to add the alias for **LFN** The LFN alias to add
Options:	
Example:	`edg-replica-manager --vo=wpsix addAlias guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a lfn:important`

COMMAND LINE TOOL

```
usage: edg-replica-manager addAlias GUID LFN
Add a new LFN alias to an existing GUID..

 Arguments:
  GUID                  The GUID to add the alias for.
  LFN                   The LFN alias to add.
```

Semantics

This command simply adds an LFN alias to an existing GUID. The LFN must be unique.

Arguments

`GUID` The GUID. It must be a valid GUID URI and must exist in the catalog.

`LFN` The new LFN. It must be a valid LFN URI and must be unique, i.e. not present in the catalog yet.

Options

none.

Return value

The call returns 0 on success (-1 on failure).

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid.
- The `GUID` does not exist.
- The `LFN` already exists.
- The RMC is unattainable or ill-configured.

3.2.7. removeAlias

Command:	**removeAlias**
Description:	Remove an alias LFN from a known GUID
Arguments:	**GUID** The GUID to remove the alias from **LFN** The LFN alias to add
Options:	
Example:	`edg-replica-manager --vo=wpsix removeAlias guid:73e16e74-26b0-11d7-b1e0-c5c68d88236a lfn:important`

COMMAND LINE TOOL

```
usage: edg-replica-manager removeAlias GUID LFN
Remove an LFN alias to GUID mapping.

 Arguments:
  GUID                  The GUID to remove the alias of.
  LFN                   The LFN alias to remove.
```

Semantics

This command removes an LFN alias from an existing GUID.

Arguments

`GUID` The GUID. It must be a valid GUID URI and must exist in the catalog.

`LFN` The LFN. It must be a valid LFN URI and must exist in the catalog.

Options

none.

Return value

The call returns 0 on success (-1 on failure).

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid.
- The `GUID` does not exist.
- The `LFN` does not exist.
- The RMC is unattainable or ill-configured.

3.2.8. printInfo

Command:	**printInfo**
Description:	Print the information needed by the Replica Manager to screen or to a file.
Arguments:	none
Options:	**-f** or **–file** *File to print the info to.*
Example:	`edg-replica-manager --vo=wpsix printInfo`

COMMAND LINE TOOL

```
usage: edg-replica-manager printInfo [options]
Print the information needed by the Replica Manager to screen or to a file.
 -f,--file <file>   The file to print the info to. This file can be used as
                    a properties file for the Stub Info system, see config options.
```

Print information about:

- replication services (RMC, LRC, ROS) and their URLs
- Information Service used (R-GMA, MDS, local configuration file)
- Storage Elements and Computing Elements with some of their attributes

This command can be used for debugging as well as for retrieving information on which storage resources should be accessible by the replica manager.

3.2.9. `getVersion`

Command:	**getVersion**
Arguments:	none.
Options:	none.
Example:	`edg-replica-manager getVersion`
Return Values:	Client version = 1.8.1

COMMAND LINE TOOL

```
usage: edg-replica-manager getVersion
Retrieve the version of both the server and the client.
```

Get the version of the replica manager client.

3.2.10. `validate`

Command:	**validate**
Arguments:	**file** The file to validate. Either one of the options must be specified as well.
Options:	**--guid** **--lfn** **--surl**
Example:	`edg-replica-manager validate lfn:myfile --lfn`
Return Values:	The file itself (for SURLs they are 'normalized', i.e. /../ and // are resolved

COMMAND LINE TOOL

```
usage: edg-replica-manager validate file [command-options]
Validate the correctness of a GUID, LFN or SURL. This is only a simple
check whether the given file adheres to the simple standard. There is no
```

```
plugin for a user-definable policy yet.
    --guid   File should be a valid GUID
    --lfn    File should be a valid LFN
    --surl   File should be a valid SURL
 Arguments:
  file                 The file to validate. Either one of the three
```

Validate a file for correctness according to the defined standards.

3.3. Optimization Commands

The catalog commands are:

- `listBestFile`
- `getBestFile`
- `getAccessCost`

3.3.1. `listBestFile`

Command:	**listBestFile**
Description:	Return the SURL that has the smallest access cost from the local store (or from the store specified by the -d option).
Arguments:	**LFNorGUID** The LFN or GUID for which we want to list the 'best' SURL as seen from the destination specified by the -d option (or the local SE if omitted)
Options:	**-d** or **--destination** *SE host* **-t** or **--turlProtocol** *protocol of returned TURL*
Example:	`edg-replica-manager --vo=wpsix listBestFile lfn:important`

Command Line Tool

```
usage: edg-replica-manager listBestFile LFNorGUID [options]
Return the SURL that has the smallest access cost from the local store (or
from the store specified by the -d option).
 -d,--destination <host>   The destination SE. The file to be found is the
                           best source for a copy to this destination. If
                           it's omitted, the local SE is used.
 -t,--turlProtocol <protocol> The protocol for the TURL that will be
                              returned. If not specified, the SURL will
                              be returned.
 Arguments:
  LFNorGUID             The LFN or GUID for which we want to list the
                        'best' SURL as seen from the destination specified
                        by the -d option (or the local SE if omitted)
```

List the best replica, i.e. the replica which can be transferred to the local SE (or the SE specified by -d) in the fastest time.

Semantics

First the LFN or GUID specified is resolved into a list of SURLs as described in the `listReplicas` command above. Then the Replica Optimization Service is contacted with the given list of SURLs, and the specified destination host (or the local SE host if none was specified). The ROS will return the SURL that has the smallest access cost from the given host at this point in time.

Arguments

`LFNorGUID` The LFN or GUID of the file of which the 'best' replica should be found.

Options

`destination` The host that is the point of reference for calculating the network and storage access cost. If not specified, the localSE is used as found through the information providers.

`turlProtocol` The protocol for the TURL that will be returned. If not specified, the SURL will be returned.

Return value

The call returns 0 on success and -1 on failure. It prints the 'best' SURL on success to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID` or `LFN` is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.3.2. `getBestFile`

Command:	**getBestFile**
Description:	Make a file available on local storage (or on the store specified by the -d option).
Arguments:	**LFNorGUID** The LFN or GUID for which we request a copy to be present at the destination specified by the -d option (or the local SE if omitted)
Options:	**-d** or **--destination** *SE host* **-p** or **--protocol** *protocol used to copy the file* **-n** or **--streams** *number of streams* **-t** or **--turlProtocol** *protocol of returned TURL*
Example:	`edg-replica-manager --vo=wpsix getBestFile lfn:important`

COMMAND LINE TOOL

```
usage: edg-replica-manager getBestFile LFNorGUID [options]
Make a file available on local storage (or on the store specified by the -d
option).
 -d,--destination <host>      The destination SE. The file to be found is the
                              best source for a copy to this destination. If
                              it's omitted, the local SE is used.
 -n,--streams <#streams>      The number of parallel streams to use. Defaults
                              to 8.
```

```
-p,--protocol <protocol>     The protocol to be used for an eventual
                             transfer. Defaults to gsiftp.
-t,--turlProtocol <protocol> The protocol for the TURL that will be
                             returned. If not specified, the SURL will
                             be returned.

Arguments:
 LFNorGUID             The LFN or GUID for which we request a copy to be
                       present at the destination specified by the -d
                       option (or the local SE if omitted)
```

Make the file available from local storage (or the storage specified by the destination option). The replica manager will find the best source to copy the file from, or will exit immediately if the file is already available on the given storage.

Semantics

First the LFN or GUID specified is resolved into a list of SURLs as described in the `listReplicas` command above. Then it is checked whether one of the replicas is actually on local storage (or the destination storage) in which case the command lists that file and exits.

If the file is not available, a listBestFile operation is carried out, with the given destination, and the best source for replication is found. Then that file is replicated to the destination just like described in the `replicateFile` command section.

Arguments

`LFNorGUID` The LFN or GUID of the file for which we need a local replica (or on the specified destination).

Options

`destination` The destination to replicate to. If not specified, the localSE is used as found through the information providers.

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

`turlProtocol` The protocol for the TURL that will be returned. If not specified, the SURL will be returned.

Return value

The call returns 0 on success and -1 on failure. It prints the SURL (or TURL) that can be used locally (or on the specified destination) upon success to stdout.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `GUID or LFN` is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.3.3. getAccessCost

Command:	**getAccessCost**
Description:	Get the access cost of the cheapest replicas for each LFN at each CE site.
Arguments:	
Options:	**-d** or **--destination** *list of CE IDs* **-l** or **--logical-file-name** *list of logical-file-names* **-p** or **--protocol** *protocol*
Example:	`edg-replica-manager --vo=wpsix getAccessCost -l lfn:one lfn:two -d lxshare0384.cern.ch:2119/jobmanager-pbs-infinite grid01.nikhef.nl:2119/jobmanager-pbs-infinite`

COMMAND LINE TOOL

```
usage: edg-replica-manager getAccessCost [options]
Get the access cost of the cheapest replicas for each LFN at each CE site.
 -d,--computing-elements <ceID>   The destination Computing Elements.
 -l,--logical-file-names <file>   The logical file names
 -p,--protocol <protocol>         The protocol to be used for transfer.
                                  Defaults to gsiftp.
```

Calculate the expected file access cost for each CE to access the set of LFNs. It makes the assumptions that no replication occurs. The `-l` and `-d` options are mandatory.

Semantics

Firstly each LFN is resolved into a list of SURLs using the listReplicas command. Then for each CE the set of best replicas for it to access is found using listBestFile for each LFN. The access cost for all the best replicas is summed up to give a total estimated access cost for each CE to access the full set of LFNs.

The access-cost array is returned, which is then printed on screen, with the best SURL for each LFN at each CE, with the corresponding estimated time to actually copy the files to local store.

Arguments

no arguments.

Options

`computing-elements` A list of valid CE IDs.

`logical-file-names` A list of valid LFNs or GUIDs.

`protocol` The protocol to be used. If it is omitted, the default protocol used, currently GridFTP.

Return value

The call returns 0 on success and -1 on failure. It prints the access cost to stdout, grouped by CEs and sub-grouped by LFNs.

Example:

```
Access Cost 0 :
CE = lxshare0286.cern.ch:2119/jobmanager-pbs-short
```

```
Best Files:
0:srm://lxshare0291.cern.ch/flatfiles/file-1
Total Access Time = 0.0

Access Cost 1 :
CE = grid01.ph.gla.ac.uk:2119/jobmanager-pbs-short
Best Files:
0:srm://lxshare0291.cern.ch/flatfiles/file-1
Total Access Time = 10.0
```

For the CE at CERN, the access cost is zero since the best replica is located on the same site. Since the best replica for the CE at Glasgow is still the replica at CERN the estimated access cost for the file according to the network information provider is 10s. Using these results, a user would be most likely to submit their job to the CE at CERN.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- An LFN is invalid or does not exist.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.4. File Transfer Commands

The file transfer commands are:

- `copyFile`
- `list`

3.4.1. copyFile

Command:	**copyFile**
Description:	Copy a file to a non-grid destination.
Arguments:	**sourceFileName** The file to copy from. This can be any kind of file: LFN, GUID, SURL, TURL or local file. **destFileName** The destination. This can only be a non-grid file - i.e. local file or TURL.
Options:	**-f** or **--force** **-p** or **--protocol** *protocol* **-n** or **--streams** *number of streams*
Example:	`edg-replica-manager --vo=wpsix copyFile lfn:mydata file:/home/bob/analysis/data5.dat`

Command Line Tool

```
usage: edg-replica-manager copyFile sourceFileName destFileName [options]
Copy a file to a non-grid destination.
```

```
-f,--force                  Overwrite the destination if it's already
                            there.
-n,--streams <#streams>     The number of parallel streams to use. Defaults
                            to 8.
-p,--protocol <protocol>    The protocol to be used for transfer. Defaults
                            to gsiftp.
Arguments:
 sourceFileName         The file to copy from. This can be any kind of
                        file: LFN, GUID, SURL, TURL or local file.
 destFileName           The destination. This can only be a non-grid file -
                        i.e. local file or TURL.
```

Copies a physical file from `source` to `destination` using the specified transport mechanism.

Semantics

This action does not involve any updates to the replica catalogs as the destination cannot be a grid-aware store, in order to avoid catalog corruption. copyAndRegisterFile needs to be used for that purpose, i.e. to bring a file into the grid. The destination does not accept GUIDs, LFNs or SURLs. As said, for those cases copyAndRegisterFile or replicateFile needs to be used.

It provides all the capability of globus-url-copy and more, accepting also GUIDs, LFNs and SURLs as the source file. Since the destination is not on grid-storage, the destination file will not be registered in the catalog.

Example: To get a local copy of a grid file, the source file can be specified as a GUID or LFN and the destination file as a local file. The result will be that the file will be copied 'out of the grid' to the local file. Local file URIs need to have the 'file' scheme.

Arguments

`sourceFileName` The source file. It may be one of the following:

- A GUID - the 'best' SE will be located to find the cheapest replica to copy from
- An LFN - the 'best' replica will be located as for the GUID
- A valid SURL, having the SE as its host name
- A valid transport URI, with a real protocol as its schema. Currently http, https, ftp and gsiftp are supported.
- A local file (specified with the 'file' schema).

`destFileName` The physical destination file. Must be one of the following:

- A local file
- A transport URI with a valid protocol. The same protocols are supported as for the source.

Options

`protocol` The protocol to be used. If it is omitted, the default protocol for the given SE is used. This parameter is only considered if the source is specified using an SURL, LFN or GUID.

`streams` The number of parallel streams to be used for the copy. If omitted, the default is used for the given protocol.

Return value

The call returns 0 on success and -1 on failure.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `source URI` is invalid or does not exist.
- The `destination URI` is invalid or not writable.
- The RLS or RMC or ROS are unattainable or ill-configured.

3.4.2. LIST

Command:	**list**
Description:	List the contents of a directory on an SRM or GridFTP server.
Arguments:	**directory** The directory or file to list.
Options:	none
Example:	`edg-replica-manager --vo=wpsix list gsiftp://testbed008.cern.ch/tmp`

COMMAND LINE TOOL

```
usage: edg-replica-manager list directory
List the directory contents on an SRM or a GridFTP server.

 Arguments:
  directory             The directory to list. This is a URI with either
                        'srm' or 'gsiftp' scheme, followed by the SE host,
                        and a proper path
```

List all the files in a specific directory, or check the existence of a file by listing it.

Semantics

The command uses the GridFTP list function and thus can be used for two things: first, for listing all files in a given directory and second for checking if a file exists. In the first case, all filenames are returned. In the second case, the filename including the path on the SE is return.

Arguments

`directory` The directory or filename to be listed or checked.

Options

none.

Return value The call returns 0 on success and -1 on failure. If the directory of file is not found, in addition a `?` is printed to the screen.

Errors

The call can fail for many reasons. Some of the reasons are:

- The user has no/expired credentials.
- The `URI` (i.e. the directory) is invalid or does not exist.

3.5. LOGGING

By default, the command line tools do *not* produce any log file, i.e. error logging is switched off. Error logging can be helpful for debugging reasons or if you want to get additional information on the error messages that are printed on the screen. You can influence logging on using the command-line switches `--log-debug`, `--log-info`, `--log-off`. The default is `--log-off`. Info level logging outputs some useful information during the various stages of each command, while debug level logging will provide far more details.

If you enable info or debug logging, the log file will be created in `/tmp/edg-replica-manager-USER.log` where USER is your username.

4. QUICK START AND SIMPLE HOW TO

In this section we give a few practical examples that can be directly applied to the EDG testbed. We assume that the user has a valid Grid proxy certificate.

Since all replication tools use the Grid security infrastructure, we first need to make sure that we have a valid user proxy. To check this, use the following command:

```
[hst@testbed010] grid-proxy-info -all
subject  : /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger/CN=proxy
issuer   : /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger
type     : full
strength : 512 bits
timeleft : 11:59:45
```

In case the user proxy is not available or has expired, renew it by issuing

```
[hst@testbed010] grid-proxy-init
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger
Enter GRID pass phrase for this identity:
Creating proxy ............................................. Done
Your proxy is valid until Sat Oct 19 04:44:30 2002
```

4.1. FIRST STEPS

In order to get familiar with the way the command line tool has to be used, you can try the basic replica manager command to retrieve the version of the client program you are using. The output of the command is given, too.

```
[hst@lxshare0313] edg-rm getVersion
Client version = 1.8.1
```

As a next step, you might want to get information about the resources in the testbed that are available to you. Here, we also remind you that you need to specify your VO (here, we assume wpsix) in the order as given below. We do not list the entire output of the command.

```
[hst@lxshare0313] edg-rm --vo=wpsix printInfo
VO used            : wpsix
default SE         : pcrd24.cern.ch
default CE         : lxshare0313.cern.ch
Info Service class : org.edg.data.reptor.info.InfoServiceStub

RMC endpoint       : http://lxshare0342.cern.ch:8080/edg-replica-metadata-catalog \
                        /services/edg-replica-metadata-catalog
LRC endpoint       : http://lxshare0344.cern.ch:8080/edg-replica-location \
                        /services/edg-local-replica-catalog
ROS endpoint       : http://lxshare0343.cern.ch:8080/edg-replica-optimization \
                        /services/edg-replica-optimization

List of CE ID's     : lxshare0313.cern.ch
```

```
                    ccgridli01.in2p3.fr
                    gppce06.gridpp.rl.ac.uk
                    ce01.nikhef.nl
                    testbed001.cnaf.infn.it
                    grid01.ph.gla.ac.uk
[...]
SE at RAL :
                     name : RAL
                     host : gppse06.gridpp.rl.ac.uk
                     type : disk
                      VOs : wpsix
             VO Directory : wpsix : /flatfiles/06/wpsix
                protocols : gsiftp
```

4.2. REPLICATION EXAMPLES USING OPTIMIZATION FUNCTIONS

This example runs through a scenario how the edg-replica-manager commands might be used to achieve standard data management tasks. In the scenario the user knows that there is a file available at CERN, that has been put on a host accessible through GridFTP. It is not a grid-aware store, so first the user has to copy the file to a Storage Element and register it in the Grid. Say that for some reason the user cannot copy it to the local CERN Storage Element but has to copy it to the one at IN2P3.

In the example the file is called 'higgs0' and resides at `testbed008.cern.ch/tmp/`.

The copy and registration is an atomic operation. In the example we assign also a Logical File Name alias to it in the process, lfn:higgs, which is easier to remember than the GUID that is returned by the call:

```
edg-rm --vo=wpsix copyAndRegisterFile gsiftp://testbed008.cern.ch/tmp/higgs0
-l lfn:higgs -d ccgridli02.in2p3.fr
```

A GUID is created and returned to the screen:

```
guid:7c29f32b-4964-11d7-a86c-9ee9a33b1f19
```

To verify whether the operation was successfully executed, we can issue listReplicas:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

which yields:

```
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
```

In order to retrieve the GUID based on the LFN, we can issue

```
edg-rm --vo=wpsix listGUID lfn:higgs
```

As a second step, the user might want to have a replica of this data file available at NIKHEF, because he intends to share it or to submit jobs that require resources at NIKHEF. A replica can be created using the replicateFile command:

```
edg-rm --vo=wpsix replicateFile lfn:higgs -d se01.nikhef.nl
```

The command confirms its execution by returning the actual SURL used. If the -d option only specifies a hostname and not a full path, an automatic SURL is created. Here the output is:

```
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

To list all replicas now in the system, we can issue listReplicas again:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

which yields:

```
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

To see which replica has the best network connection to CERN, we can use listBestFile:

```
edg-rm --vo=wpsix listBestFile lfn:higgs -d pcrd24.cern.ch
```

The output is, for example:

```
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
```

which means that the file at NIKHEF can be made available at CERN faster than the one from LYON. We now want to see the file access costs of the best replica with respect to CERN, NIKHEF and Lyon using a CE hostname from each site:

```
edg-rm --vo=wpsix getAccessCost -l lfn:higgs \
       -d lxshare0313.cern.ch:2119/jobmanager-pbs-short\
       ce01.nikhef.nl:2119/jobmanager-pbs-short\
       ccgridli01.in2p3.fr:2119/jobmanager-pbs-short
```

The output is:

```
Access Cost 0 :
CE = lxshare0313.cern.ch:2119/jobmanager-pbs-short
0:srm://se01.nikhef.nl/flatfiles/wpsix/higgs
TotalTime = 0.13

Access Cost 1 :
CE = ce01.nikhef.nl:2119/jobmanager-pbs-short
0:srm://se01.nikhef.nl/flatfiles/wpsix/higgs
TotalTime = 0.0

Access Cost 2 :
CE = ccgridli01.in2p3.fr:2119/jobmanager-pbs-short
0:srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
TotalTime = 0.0
```

The list is grouped by the CEs given on the command line. For each CE the 'best' replica is listed and the time it would take to make it available locally. We can see that the expected access cost to transfer the file from NIKHEF to CERN is 0.13 sec. For the other sites the access cost is 0 since the file is already locally available and no network transfer is required.

To actually make the best file available at CERN, we can issue getBestFile:

```
edg-rm --vo=wpsix getBestFile lfn:higgs -d pcrd24.cern.ch
```

The output is something like:

```
srm://pcrd24.cern.ch/data/temp/a6289c7c-4966-11d7-bc63-d91230733e2d
```

We should now have three replicas:

```
edg-rm --vo=wpsix listReplicas lfn:higgs
```

The output is:

```
srm://pcrd24.cern.ch/data/temp/aaa64014-4967-11d7-a6cc-f7a1ff1899b0
srm://se01.nikhef.nl/flatfiles/wpsix/higgs
srm://ccgridli02.in2p3.fr/edg/StorageElement/dev2/wpsix/higgs
```

To delete a replica on a specific SE we can use the deleteFile command with the -s option:

```
edg-rm deleteFile lfn:higgs -s ccgridli02.in2p3.fr
```

5. TEST SUITE

The RPM edg-replica-manager-1.8.1-1-test.rpm contains two test suites, one for the API (see 5.1. and one for the command line tools 5.2.). These test suites test all possible commands that are provided by the Replica Manager.

5.1. API TESTS

Currently, there is one API test suite which tests the Java API.It is located at:

```
$EDG_LOACTION/sbin/test/edg-replica-manager-testJavaAPI
```

The test suite tests each of the API methods and provides a small integration test with the following services:

- Replica Location Service
- Replica Metadata Catalogue
- Replica Optimization Service
- Information Service (correct class has to be selected in the file edg-replica-manager.conf)
- Storage Elements

USAGE

```
Usage: edg-replica-manager-testJavaAPI [ -v ] <vo-name> [ <se1> <se2> ]
       -v,--verbose print more verbose test information
       <vo-name>    Virtual Organization
       <se1>        Storage Element1: optional parameter. If not given,
                    it will be determined from the Information Service.
       <se2>        Storage Element2: optional parameter. If not given
                    it will be determined from the Information Service.
```

If you use the verbose mode, you get all details about what commands are executed. In addition, all tests are logged in the log file of the EDG Replica Manager if logging is switched on by default (the command-line logging options are not supported).

EXAMPLE

```
> edg-replica-manager-testJavaAPI cms
Start Replica Manager test suite

test copyFile
test registerFile
test listReplicas
test replicateFile
test listBestFile
test getBestFile
test getTurl
test deleteFile
```

```
test copyAndRegisterFile
test copyAndRegisterFile
test listGUID
test addAlias
test listGUID
test removeAlias

Test Result
===========
copyFile             : OK
registerFile         : OK
listReplicas         : OK
replicateFile        : OK
listBestFile         : OK
getBestFile          : OK
getTurl              : OK
deleteFile           : OK
copyAndRegisterFile  : OK
listGUID             : OK
addAlias             : OK
removeAlias          : OK

Integration tests
  - Interaction with Replica Location Service     : OK
  - Interaction with Replica Metadata Catalogue   : OK
  - Interaction with Replica Optimization Service : OK
  - Interaction with Storage Elements             : OK
  - Interaction with the Information Service      : OK

Total tests run: 14; failed: 0; skipped: 0
```

5.2. COMMAND LINE TESTS

The command line test suite tests each of the tools provided by the EDG Replica manager. It is located at:

```
$EDG_LOACTION/sbin/test/edg-replica-manager-testCLI
```

Note that usually the API test takes shorter time to execute since the Java Virtual Machine is only started once. However, in order to test your entire installation, you should run both test suites.

USAGE

```
Usage: edg-replica-manager-testCLI
        --vo=voname
       [--edg-rm-executable=path + name of the edg-rm executable]
       [--test=path + name of a single test to run]

Example: edg-replica-manager-testCLI --vo=wpsix \
        --edg-rm-executable=/home/user/edg-reptor/dist/bin/edg-rm \
        --test=cli-func-tests/edg-rm-getAccessCost-test
```

All standard output and error is written to a log file of the form

```
/tmp/edg-reptor-testcli-<user id>-<process no>.log
```

Each individual script has three possible return codes:

- 0 - successful test.
- 1 - an error occurred somewhere other than in the command under test (e.g. in the setting up or cleaning up part of the test).
- 2 - an error occurred in the command under test.

edg-replica-manager-testCLI gives a one-line status report for each test, reflecting the return status, to standard out. For more information on errors the log file should be consulted.

For information on SEs and CEs the edg-rm printInfo command is used, so this command is tested first and if it fails, none of the other tests can be carried out.

EXAMPLE

```
> edg-replica-manager-testCLI --vo=cms

 edg-rm printInfo test: this must pass to run the rest of the test suite
 4 CEs and 13 SEs found: printInfo test passed

edg-rm cli tests started - output in /tmp/edg-replica-manager-testcli-46221-4556.log

edg-rm-addAlias-test : OK
edg-rm-copyAndRegisterFile-test : OK
edg-rm-copyFile-test : OK
edg-rm-deleteFile-test : OK
edg-rm-getAccessCost-test : OK
edg-rm-getBestFile-test : OK
edg-rm-getVersion-test : OK
edg-rm-listBestFile-test : OK
edg-rm-listGUID-test : OK
edg-rm-listReplicas-test : OK
edg-rm-registerFile-test : OK
edg-rm-removeAlias-test : OK
edg-rm-replicateFile-test : OK

edg-rm cli tests finished
```

6. APPENDIX: REPLICA MANAGER FOR EDG RELEASE 2 VERSUS EDG RELEASE 1

The current EDG Release 2 Replica Manager has some functionality changes from the previous EDG Release 1 version (first released in May 2002) due to some important design changes. In this section we describe a few differences between the two replica managers in order to provide an easier start for people already familiar with the EDG Release 1 `edg-replica-manager`. Whenever we use the program name `edg-replica-manager` we refer to the old tool, and we use the term `edg-rm` or EDG Replica Manager to refer to the new replication tool.

Basically, the new interface is very similar to the old one. However, there are more features added and more Grid services used. The following list outlines the main differences.

- `edg-rm` uses the **Replica Location Service (RLS)** and the **Replica Metadata Catalog** and thus there is no restriction of the use of Logical File Names. The LFN can be any user supplied string prefixed by `lfn:`. In addition, alias names are allowed for LFNs.
- In addition to LFNs, **GUIDs (Grid Universal Identifiers)** are uses to uniquely identify Logical File Names.
- For all physical filenames or site filenames, a **protocol needs to be added as a prefix**. For `edg-replica-manager` the hostname alone was sufficient but `edg-rm` also requires the protocol name in the file name. Example:

  `edg-replica-manager` (old):

  ```
  copyAndRegisterFile -s host1.cern.ch/home/data/testfile
  ```

  `edg-rm`:

  ```
  copyAndRegisterFile gsiftp://host1.cern.ch/home/data/testfile
  ```

  Also the protocol ”file:” is allowed if a file is available locally.
- The command line interface (CLI) has changed slightly: instead of having an executable for each method (edg-replica-manager), `edg-rm` provides a single executable where the method is passed as the first parameter.

  ```
  edg-rm copyAndRegister
  ```

  rather than

  ```
  edg-replica-manager-copyAndRegister
  ```

  `edg-rm` has one **Java** rather than several C++ executables.
- No direct MSS interface: The new Replica Manager uses the SRM for that purpose
- The EDG Replica Manager package is a pure Java package and does not provide a C++ interface to the end user. A limited C++ API is provided for the methods listReplicas and getAccessCosts and is part of the edg-replica-manager-client package (C++ based).
- The new replica manager uses the Replica Optimization Services (ROS, also called Optor) and thus provides several replica optization methods like getBestFile, listBestFile, getAccessCost etc.
- registerFile/unregisterFile instead of registerEntry/unregisterEntry

REFERENCES

[1] Leanne Guy, Erwin Laure, Peter Kunszt, Heinz Stockinger, Kurt Stockinger. Replica Management in Data Grids. Technical report, Global Grid Forum Informational Document, GGF5, Edinburgh, Scotland, July 2002.
http://edms.cern.ch/document/350430

[2] WP2 DataManagement, WP2 Replica Manager Design Specification, Draft 0.7, 21 March 2002.
http://edms.cern.ch/document/338668/

[3] Ann Chervenak, Ewa Deelman, Ian Foster, Wolfgang Hoschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Heinz Stockinger,Kurt Stockinger, and Brian Tierney. Giggle: A Framework for ConstructingScalable Replica Location Services. In Proc. of the Int'l. IEEE Supercomputing Conference (SC 2002), Baltimore, USA, November 2002.

[4] RLS: http://cern.ch/edg-wp2/replication/replica-location-service/

[5] EDG Replica Manager: http://cern.ch/edg-wp2/replication/

[6] R-GMA Documentation: http://hepunx.rl.ac.uk/edg/wp3/documentation/index.html

[7] W. H. Bell, D. G. Cameron, L. Capozza, P. Millar, K. Stockinger, F. Zini. Design of a Replica Optimisation Framework. Technical Report DataGrid-02-TED-021215, CERN, Geneva, Switzerland, December 2002.

[8] Ceki Gülcü. Short introduction to log4j. http://jakarta.apache.org/log4j/docs/manual.html