

Data Management

Data Management

Data management on the grid takes place on two different levels:

Physical level: Storage Element

Logical level: Catalogue

The Storage Elements (SE) are the computers that physically hold the file data. Access to them is provided through different access protocols, the most common being the Grid File Transfer Protocol (GridFTP²). A Logical File Name (LFN) is used to refer to files in a location-independent way; one or more LFNs² can be associated to a physical file on a SE. LFNs² are saved in a catalog, which also contains the metadata about the files the logical filenames point to, like: real name, dimension, access permissions, modify time, etc.; thus giving the user the illusion of a single Grid-wide UNIX-like file system.

Since the physical files can reside on different SEs², whereas all the logical names are in the same catalog, the LFN provides to the Grid user the illusion of working on a single large distributed file system. To use a file, it is sufficient to know its logical filename and not the details about its location.

The LCG File Catalog

The LCG File Catalog (LFC) server and its associated client programs (installed among the UI software) implement a Grid-level UNIX file system look-alike. In particular, LFC organizes LFNs² in a hierarchical directory structure, and features POSIX-like ACLs² to protect access to LFNs².

You can operate with LFNs² in LFC just as you operate on filenames in your local UNIX/Linux file system. Indeed, LFC commands (you can tell them by the prefix `lfc-` in the name) are similar in operation to their UNIX counterparts: for instance, `lfc-ls` lists LFNs² in the same way that UNIX `ls` lists local filenames.

Storage Elements

Storage Elements (SEs²) are the computers where Grid-accessible files are actually stored. At present, the EGEE/LCG Grid is in between a transition from the older "`Classic SE`" software to the newer "`SRM`" software.

A "`Classic SE`" is nothing more than a disk server equipped with a GridFTP² server. GridFTP² is an extension to the FTP protocol, allowing authentication with Grid credentials (that is, a proxy certificate).

"`SRM`" aims at providing a uniform interface for all kinds of Grid data storage, most notably large tape-based systems and disk servers. With the SRM protocol, a client may request access to a certain file, reserve space for putting files in the SE managed storage, or schedule a copy from an SE to another. It should be noted that SRM is not a data access protocol; all data transfers from the SE to the client (or vice versa) will take place through some other protocol (GridFTP², RFIO, HTTP, etc.).

A Glossary of Grid Data Management

Storage URL

A storage URL is a URL exactly locating a copy of a Grid-stored file; a SURL encodes information about the SE physically holding that copy, and a path to refer to the data within that SE's namespace. Examples of SURLs_? are:

```
sfn://egrid-3.egrid.it/storage/grid/gridats/crossi/Myfile.test
srm://stormse.grid.elettra.trieste.it/gridcc/test1.txt
```

The difference between `sfn://` and `srm://` type SURLs_? is only connected to the kind of protocol used to access the data, and is handled transparently by the middleware; it should be of no concern to the end user.

Logical File Name

The Logical File Name (LFN) is a string tagging a Grid-stored file, independently of the SE where it is infact stored. An LFN takes the form of a UNIX path, beginning with `/grid`; for instance:

```
lfn:/grid/gridats/hello.txt
```

Note

Note:

The syntax is a bit inconsistent, in that some commands will reject the `lfn:` prefix.

In other words, an LFN is a nickname for a set of SURLs_?, all referring to copies of the same Grid file, residing on different SEs_?. LFNs_? can be up to 254 characters long.

Grid Unique Identifier

A GUID is a long hexadecimal string (representing a 128 bit UUID) that the LFC software uses internally to tag non-alias LFNs_?. Although GUIDs_? are still exposed by the software interface for backwards compatibility, they are no longer essential, and all common data management operations can be performed using LFNs_? and SURLs_?.

Alias

An alias is to a LFN (in LFC) what a symbolic link is to a filename (in UNIX file system). In other words, an alias is a kind of LFN associated with another LFN instead of a (set of) SURL.

For all data management purposes, an alias behaves exactly like a "regular LFN" (you cannot tell a regular file from a symbolic link with the standard UNIX commands), so we shall use the term LFN to refer to both.

A more in-depth analysis of the relation between these: in the LFC, any regular LFN is associated with a GUID (and this relation is bijective); any GUID is associated with SURLs_? in a one-to-many relation; finally, each alias is associated with a regular LFN, and a regular LFN may be the target of many aliases. Summing up, we actually have a many-to-many relationship among LFNs_? and SURLs_?.

There is an explicit and strict parallel between LFC and the UNIX file system, which is actually reinforced by the `lfc-*` commands and their UNIX counterparts; indeed, the following correspondence works quite exactly:

LFC	UNIX file system
LFN	File name
GUID	Inode number
alias	Symbolic link

Since aliases and "real" LFNs exhibit the same behavior in all data management operations, we shall use the term LFN in the sequel to mean either a "real" LFN or an alias.

Operating With Grid-Stored Files

The main operations that can be performed on Grid-stored files are:

[Listing files in the Catalogue](#)

[Uploading local files to the Grid](#)

[Downloading a Grid-stored file](#)

[Removing a file](#)

[Renaming a File](#)

[Creation and Deletion of Directories](#)

[Linking more than one logical filename to the same file](#)

Commands operating only on the LFC Catalog are of the form "lfc-", and they use the environment variable *LFC_HOST* to get the address and port of the catalog they work with; whereas the commands that operate on data stored on a SE have the form *lcg-* and require an option `--vo` followed by the name of the Virtual Organization (VO) the user belong in.

Listing Files in the Catalogue

The `lfc-ls` command shows the content of a specific directory in the catalogue. It takes a single option and a parameter, which will be the LFN of the directory stripped of the `lfn:/` prefix. Therefore, to inspect the content of the directory `lfn://grid/gridats` the command will be:

```
bash$ lfc-ls /grid/gridats
Datadir
data
alias
```

The option `-l` works like the corresponding one in the UNIX `ls` command and show more detailed information about the file, thus letting the user distinguish between aliases, directories and regular files (and read other information as well).

```
bash$ lfc-ls -l /grid/gridats
drwxr-xr-x 1 506 504    0 Aug 17 10:30 Datadir
-rwxrwxr-x 1 506 504 1645 Aug 10 17:57 data
lrwxrwxrwx 1 506 504    0 Aug 10 18:43 alias -> /grid/gridats/data
```

An alias is seen as a symbolic link by the command `lfc-ls -l` and is a logical filename which is linked to a file which already has a LFN (also see `lfc-ln`, `lcg-aa` in section [linking more than one logical filename to the same file](#)).

Uploading Local Files to the Grid

To copy a file on the Grid it is necessary to know at least the name of one of the SEs; how to achieve this is explained in the section 3.2.1. File can be uploaded using the `lcg-cr` command. The command has the following format and meaning of attributes and options are listed in table 5.1:

```
lcg-cr -d "destination" --vo "VO" [-l "lfn"]? "src_file"
```

The `lcg-cr` command takes care of copying the file on the SE and (if `-l` is specified) registers it on the catalog.

Table 5.1 – Attributes of `lcg-cr` command

<code>-d</code>	Either a SURL or the hostname of the SE. To get the list of the available SE, use the command <code>egrid-infosites</code> . (Mandatory)
<code>-l</code>	This option is not mandatory. If not specified, the file will not get registered
<code>--vo</code>	As usual, this parameter specify the virtual organization.
"src_file"	This is the source file. The access protocol protocol (chosen between <code>file:</code> and <code>gsiftp:</code>) must be prefixed.

Step 1

Let us search the name of a valid SE:

```
bash$ lcg-infosites --vo gridats se
Avail Space(Kb) Used Space(Kb) Type SEs
-----
685648064      876802208      n.a grid002.oat.ts.astro.it
13659804       1054456         n.a egrid-ce-01.egrid.it
169452728      7753952         n.a gridts01.grid.elettra.trieste.it
```

Step 2

We notice that `egrid-ce-01.egrid.it` is available and still has some free disk space: now we are ready to copy and register a file, which in case of success will output the GUID of the newly created Grid file:

```
bash$ lcg-cr --vo gridats \
-l lfn:/grid/gridats/test.txt \
-d egrid-ce-01.egrid.it \
file:/home/crossi/test.txt
guid:298c666e-9792-4a69-9c5a-6594d755b3eb
```

Step 3

We can now check that the file has been copied and registered by looking at the name in the catalog (since we specified the `-l` option):

```
bash$ lfc-ls -l /grid/gridats/test.txt
-rwxrwxr-x 1 60125 60125 0 Aug 22 14:52 /grid/gridats/test.txt
```

Downloading a Grid-Stored File

Knowing the logical name of a file is the only information we need in order to get a file from the Grid that has been registered in the LFC catalogue. Note that in the near future this could be the only way of dealing with files, though it is still possible to use the GUID or SURL for non-registered files. For this purpose, the `lcg-cp` command can be used which is having the following format:

```
lcg-cp --vo "VO" "grid_file" "local_file"
```

The `--vo` defines the virtual organization which `"grid_file"` and `"local_file"` define the source file and destination file. Value of `"grid_file"` should be specified with the corresponding protocol (`lfn:`, `guid:`, `sfn:` or `srm:`) and `"local_file"` should be composed of the `file:` prefix followed by the *absolute* path to the file. Following example illustrates the use of `lcg-cp` command:

```
bash$ lcg-cp --vo gridats lfn:/grid/gridats/pippo file:$(pwd)/pippo
```

Removing a File

To delete both a physical copy of a file and its logical name in the catalog the command `lcg-del` should be used. The syntax is:

```
lcg-del --vo "VO" [-a | -s "SE"]? "grid_file"
```

Table 5.2 – Attributes of `lcg-del` command

<code>--vo</code> "VO"	Name of the virtual organization
<code>-a</code>	Remove all the replicas of the file (mutually exclusive with <code>-s</code>)
<code>-s "SE"</code>	Remove only the replica residing on the specified SE (mutually exclusive with <code>-s</code>)
"grid_file"	Specify the Grid file to delete; must specify the protocol too, which can be <code>lfn:</code> , <code>guid:</code> , <code>sfn:</code> or <code>srm:</code>

For example, let us assume this is the situation in the catalog:

```
bash$ lfc-ls -l /grid/gridats/testdir
drwxr-xr-x 1 506 504 0      Aug 17 10:30  Datadir
-rwxrwxr-x 1 506 504 1645 Aug 10 17:57  file1
-rwxrwxr-x 1 506 504 1645 Aug 11 17:57  test
```

and we want to remove the file `test` on every SE which contain it (all replicas). It will be sufficient to issue the command:

```
bash$ lcg-del -a --vo gridats lfn:/grid/gridats/testdir/test
```

Indeed:

```
bash$ lfc-ls -l /grid/gridats/testdir
drwxr-xr-x 1 506 504 0      Aug 17 10:30  Datadir
-rwxrwxr-x 1 506 504 1645 Aug 10 17:57  file1
```

Renaming a File

The `lfc-rename` command can be used to rename a file or directory in the catalog. The syntax of this command is:

```
lfc-rename "old_path" "new_path"
```

where `"old_path"` and `"new_path"` are, respectively, the name of the file (or directory) before and after the rename operation. For example:

```
bash$ lfc-rename \
    /grid/gridats/testdir/pippo.txt \
    /grid/gridats/testdir/pluto.txt
```

This command works only at the logical level (LFN) and thus is an action that is performed on the catalogue only. Renaming of storage-level paths (i.e., SURLS²) is not supported, as it would entail performance and integrity issues on the LFC.

Creation and Deletion of Directories

To create a directory in the LFC catalog, the command `lfc-mkdir` is used, with the syntax:

```
lfc-mkdir [-m numeric_mode] [-p] "path"
```

The `-m` is optional, it specifies the permissions in the same way as the `chmod` command on UNIX, and the default value is `0777`. The `-p` argument is optional and if used it creates a parent directory (e.g. to create a directory `/grid/VO/one/two/tree` in one step, without having to create the directory `one/` and `two/` first). The path defines the logical filename of the directory (*without* the `lfn:` prefix).

To remove directories (which do not physically exist on any SE, as stated before) use `lfc-rm` command having the syntax:

```
lfc-rm -r path
```

where `path` is the name of the directory (without the `lfn:` prefix).

As said before, it is possible to rename a directory in the same way as a file via the command `lfc-rename`.

Linking more than one Logical Filename to the Same File

It is possible to associate more than one LFN to a single file; the LFNs² beyond the first one are called 'aliases'. Two commands are available to perform this: `lcg-aa` and `lfc-ln`. The `lfc-ln` command syntax is very simple and having the following form:

```
lfc-ln -s "original_name" "new_alias or directory"
```

Where `"original_name"` defines the logical filename to which the alias refers to (without `lfn:`). The `"new_alias"` or `directory` allows the alias to be created by specifying the complete pathname or simply the directory in which it will be created (which will generate an alias with the same name of the original, if another file with the same name is not present). This is analogous to the UNIX `ln -s` command.

```
bash$ lfc-ln -s /grid/foo /grid/baz
```

The syntax of the command `lcg-aa` is slightly more complex:

```
lcg-aa --vo "vo" "guid" "lfn"
```

Where `--vo` defines the virtual organization and `"guid"` specify the GUID of the file. Here the `lfn:` prefix is mandatory.

Example:

```
bash$ lcg-aa --vo gridats \  
  guid:b0f64a56-0f1e-4ed2-993e-631de0bd06bb \  
  lfn:/grid/gridats/testjpb_rf12
```

Advanced Commands

This section contains information about commands which require a slightly higher skill than the former ones and allow a finer grained control on the data on the Grid.

Aside from the basic operation, it is also possible to show and modify the auxiliary information about the single files (metadata), customize in a more specific way the access permissions to them and create more replicas of a single file.

Getting Information about the Access List

UNIX-like permissions are not the only security information associated with the LFNs: Access Control Lists (ACL) are also presents, which can be shown and changed. The ACLs specify who can access a specific file or directory and which kind of permissions he/she has (read/write). The ACLs on a LFN do not affect the retrieval/modification permits of "data" in a Grid-stored file; they merely determine the retrieval/modification of SURLs (i.e., Grid storage location pointers) associated with that LFN. Moreover, the ACL for the directories are also used to change the permissions of all the files that are inside them.

The command `lfc-getacl` shows the ACL of a file or directory. It has the following format:

```
lfc-getacl "path"
```

The result will be something like:

```
bash$ lfc-getacl /grid/gridats/testdir/testfile
# file: /grid/gridats/testdir/testfile
# owner: 60125
# group: 60125
user::rwx
group::rwx #effective:rwx
other::r-x
```

The first three lines show the name of the file and the user and owner of it respectively. Every file is associated with an ACL, which will be a default value, like the one shown above, unless the user does not change them, directly or by changing the ACL of the parent directory, that contains the file.

Every line that does not start with a # sign specify an ACL rule which can be applied to a single user (lines which start with `user:`), a single group (lines which start with `group:`) or to every user which is not contemplated in the other rules (`other:`). The lines which start with `user:` or `group:` can also be used to specify access control to a specific group or user. For more information, please consult the Linux man pages `getfacl(1)`, `setfacl(1)` and `acl(7)`.

When issued in a directory the command `lfc-getacl` will likely show other lines too:

```
bash$ lfc-getacl /grid/gridats/testdir
# file: /grid/gridats/testdir
# owner: 60125
# group: 60125
user::rwx
group::rwx #effective:rwx
other::r-x
default:user::rwx
default:group::rwx
default:other::r-x
```

The lines which start with the `default:` keyword show the permissions that will be automatically assigned to LFNs² and subdirectories created inside the directory `/grid/gridats/testdir`.

Getting the GUID of a file

The `lcg-lg` command allows the user to retrieve the GUID of a file, given its logical name or one SURL. The syntax is:

```
lcg-lg --vo "vo" "lfn_or_surl"
```

As usual `--vo` defines the virtual organization while "lfn_or_surl" defines the file, as a logical filename (prefix `lfn:`) or SURL (prefixed with `sfn://` or `srm://`).

Let us look at an example that uses `lfn:`:

```
bash$ lcg-lg --vo egrid lfn:/grid/gridats/test.txt
guid:298c666e-9792-4a69-9c5a-6594d755b3eb
```

Let us look at another example that uses a surl:

```
bash$ lcg-lg --vo gridats \
      sfn://egrid-ce-01.egrid.it/storage/gridats/generated/2005-08-22/filec34d1b57-84be-4898-94
guid:298c666e-9792-4a69-9c5a-6594d755b3eb
```

Replica Handling and Creation

As previously mentioned every registered file is associated with a GUID although can be linked to more than a SURL and a LFN, therefore a file can reside on multiple SE (more than one SURL) and/or can be accessed through more than 1 logical name (if aliases are present).

The duplicates of files that resides on additional SE (but are accessed with the same lfn) are called ?replicas? and are created with the command `lcg-rep`. Replicas are useful especially when handling large file, to ensure that their data are available "near" (in a network sense) the computing site (the WNs² usually).

Showing the Replica of a File

The `lcg-lr` command shows a list of SURL associated with the specified file. The file can be selected using a LFN, a GUID:

```
lcg-lr --vo "VO" "file"
```

where `--vo` defines the virtual organization while file defines the file name with the protocol (`lfn:`, `guid:`).

Let us look at an example. To know the physical locations of LFN `/grid/gridats/test.txt`, type:

```
bash$ lcg-lr --vo gridats lfn:/grid/gridats/test.txt
sfn://egrid-ce-01.egrid.it/storage/gridats/.../2005-08-22/filec34...
```

If we would know the GUID instead:

```
bash$ lcg-lr --vo gridats guid:298c666e-9792-4a69-9c5a-6594d755b3eb
sfn://egrid-ce-01.egrid.it/storage/gridats/.../2005-08-22/filec34...
```

If the file have more than one replica, then the command would list them all:

```
bash$ lcg-lr --vo gridats lfn:/grid/gridats/test2.txt
sfn://baciuco.hpc.sissa.it/storage/gridats/.../2005-08-22/filea7123...
sfn://egrid-ce-01.egrid.it/storage/gridats/.../2005-08-22/filec7423...
```

Creating the Replica of a File

The command used to create a replica is `lcg-rep`, whose syntax is as follows:

```
lcg-rep --vo "VO" -d "destination" "src_file"
```

where `--vo` defines the virtual organization while "destination" is the SURL or simply the hostname of a SE. The `src_file` indicates the source file prefixed with the protocol (`lfn:`, `guid:`, `sfn:` or `srm:`).

For example, assuming that a file `lfn:/grid/gridats/test.txt` is present, which resides on `egrid-ce-01.egrid.it`:

```
bash$ lcg-lr --vo gridats lfn:/grid/gridats/test.txt
sfn://egrid-ce-01.egrid.it/storage/gridats/generated/2005-08-22/filec34d1b57
```

To create a replica on `baciuco.hpc.sissa.it` (a copy of the file that can be accessed through the same LFN) we issue the command:

```
bash$ lcg-rep --vo gridats -d baciuco.hpc.sissa.it \
      lfn:/grid/gridats/test.txt
```

The command `lcg-lr` will show us all the replicas of the file, thus proving that the operation completed successfully:

```
bash$ lcg-lr --vo gridats lfn:/grid/gridats/test.txt
sfn://baciuco.hpc.sissa.it/storage/gridats/.../2005-08-22/filea57...
sfn://egrid-ce-01.egrid.it/storage/gridats/.../2005-08-22/filec34...
```

Note

Note:

When specifying the host of the destination SE, the file has been placed into an automatically generated file (this behaviour can be changed, e.g. by simply using the full SURL of the destination).

Showing the LFNs₂ pointing to a given SURL

The command `lcg-la` allows the user to list the LFNs₂ pointing to a given GUID or SURL; the syntax is:

```
lcg-la --vo "VO" "file"
```

as usual `--vo` is the virtual organization while `file` is the file name with protocol (`lfn:`, `guid:` or SURL).

```
bash$ lfc-ls -l /grid/gridats/testdir/
lrwxrwxrwx 1 60125 60125 0 Aug 22 17:02 test-alias.txt -> /grid/gridats/testdir/test.txt
-rwxrwxr-x 1 60125 60125 0 Aug 22 17:01 test.txt
```

```
bash$ lcg-la --vo gridats lfn:/grid/gridats/testdir/test.txt
lfn:/grid/gridats/testdir/test-alias.txt
```

ELFI – EGRID LFC Filesystem Interface

ELFI is a file system interface to the LFC catalog and LCG SEs₂ (both "classic" and SRM-based). With ELFI, you can see the entries in the LFC catalog as files in a locally mounted file system, and directly operate on the replica contents: read/write operations on the local file system are acted as read/write operations on a remote SE via the GSI-RFIO protocol.

All operations on the local file system are translated into the appropriate operations on the LFC catalog or the remote SE (via RFIO protocol). All operations on the catalog or the SE have a local file system equivalent. Indeed, ELFI can be used to:

- transparently access a Grid file content by logical file name:

```
$ emacs /elfi/LFC/path/to/lfn
```

- create and register a file in the catalog:

```
$ cp /local/file /elfi/LFC/path/to/lfn
```

- delete a file from the catalog, and all of its replicas:

```
$ rm /elfi/LFC/path/to/lfn
```

- delete a single replica from the catalog:

```
$ rm  
/elfi/<SE-name>/path/to/replica
```

- replicate a file across SEs₂:

```
$ ln /elfi/LFC/path/to/lfn  
/elfi/<SE-name>/path/to/replica
```

- list an SE content:

```
$ ls -l /elfi/<SE-name>/path
```

- list LFC catalog content:

```
$ ls -l /elfi/LFC/path
```

ELFI is built on top of the File System in User Space (FUSE) system. The FUSE suite provides an interface layer by which a user-space program may respond to Linux kernel Virtual File System (VFS) requests. Any file system-like content may thus be seen as a "real" file system on Linux. FUSE consists of a kernel module and a user-space interface library; FUSE is in mainstream Linux kernel $\geq 2.6.14$, or available as an add-on module since Linux kernel version 2.4.