

---

**Joint Regional CE EGEE and SEEGRID-2  
Summer School on Grid Application Support  
[www.egee.hu/grid06](http://www.egee.hu/grid06)**



---

**Practice of EGEE data management  
Part 2 – gfal API**

**Gabor Hermann  
MTA SZTAKI**

**Goal:** Use grid files directly from a C (C++) program.

Temporary restriction: gfal calls for the time being rfio. Consequence: Because rfio is not Grid safe only grid files located on the same site (local file system) can be used.

An other unpleasant consequence due to rfio: sfn (storage location defined) files must be used in the special form  
sfn://<hostname>//<path>

The example c++ program `gfal_example.cpp` uses an arbitrary existing grid file:

Opens it for writing, fills it with a simple content and reopens it for control reading.

**Practice:**

**1. Let us copy the files** "gfal\_example.cpp" "gfal\_examplpe.jdl" "gfal\_exampleComp.sh" from a central site in our directory:

**1.a** On day 1 you have received a two digit number between 01-70. (**XY**). Use this number to login to one of the machines:

Account: budapest**XY**  
Password: GridBUD**XY**

Open a terminal window and login to the GILDA UI machine:

```
ssh budapestXY@glite-tutor.ct.infn.it  
Password: GridBUDXY
```

**1.b** Download and decompress the training material:

```
wget http://www.sztaki.hu/~ghermann/Szemelyes/SommerScool_06/SummerSchool_06.zip  
unzip -x SummerSchool_06.zip
```

**1.c** Make a new directory there

```
mkdir <workdir>  
cd <workdir>
```

```
cp ~/SummerSchool_06/GfalSources/gfal_example.cpp .
cp ~/SummerSchool_06/GfalSources/gfal_example.jdl .
cp ~/SummerSchool_06/GfalSources/gfal_exampleComp.sh .
```

## 2. Let us compile the C++ program:

### 2.a Change the access right in order to let the script to run

```
chmod +x gfal_exampleComp.sh
```

### 2.b Compile the program defined in the script:

```
./gfal_exampleComp.sh
```

## 3. Testing the compiled C program on the UI machine:

### 3.a Controlling the valid proxy:

```
voms-proxy-info
```

If its lifetime is too short

```
voms-proxy-destroy
voms-proxy-init -vo gilda
```

### 3.b Creating an arbitrary local test file:

```
history > dum
```

### 3.c Executing the test.

**In the argument absolute path is needed!**

```
./gfal_example file:<path_of_workdir>/dum
```

For example – if **XY** is 13 and your working directory is “/home/budapest13/workdir”:

```
./gfal_example file:/home/budapest13/workdir/dum
```

## 4. Testing the compiled C program in the Grid:

### 4.1 Let us make an arbitrary Grid file from the local file (See 3.b):

#### 4.1.1 Let us find a Storage Element

```
lcg-infosites --vo gilda se
```

#### 4.1.2 Let us create a grid file from "dum" using a storage found in the listing of 4.1.1

```
lcg-cr --vo gilda -d <SE> file:<path_of_workdir>/dum
```

*Please use the **grid005.iucc.ac.il** as <SE>*

For example (if your **XY** number is 13):

```
lcg-cr --vo gilda -d grid005.iucc.ac.il \
file:/home/budapest13/workdir/dum
```

This command returns the **guid** of the created grid file:  
guid:4d75ced2-df7c-44c3-901a-bd8a172ae275

#### 4.1.3 Let us determine the storage address of the grid file upon guid returned by 4.1.2

This value will be used in the "gfal\_example.jdl" file.

```
lcg-lr --vo gilda <guid_resulted_from_4.1.2>
```

For example:

```
lcg-lr --vo gilda guid:4d75ced2-df7c-44c3-901a-bd8a172ae275
```

The result may look like the following:

```
sfn://grid005.iucc.ac.il/storage/gilda/generated/2006-06-28/filef78913b6-427a-4df0-806b-c7ea48ae0a04
```

#### 4.2 Let us edit the "gfal\_example.jdl" and put the storage address as the value of "Arguments".

**!!!!!! There is one fallacy !!!!**

*Because of the implied use of "rfio" the form sfn://<hostname>//<path> must be used!*

For example:

```
>>>-----+-----+
          |           |
          v           v
Arguments="sfn://grid005.iucc.ac.il//storage/gilda/generated/2006-06-28/filef78913b6-427a-4df0-806b-c7ea48ae0a04"
```

**After coping the sfn name as "Arguments", an additional / sign must be inserted after the host name!**

#### 4.3 Let us run the program

##### 4.3.1 Let us investigate where our program may run.

The list of CE-s will be stored for example in "res.list"

```
glite-job-list-match -o res.list --vo gilda gfal_example.jdl
```

##### 4.3.2 Let us submit our program and store the job identifier in file - for example in file "job.list01"

```
glite-job-submit -i res.list -o job.list01 gfal_example.jdl
```

***Fallacy: as we use Rfio, the CE must be local to SE. Therefore the CE on the site "grid004.iucc.ac.il" must be selected!***

##### 4.3.3 Let us observe the state of the job

```
watch "glite-job-status -i job.list01"
```

If the displayed state reaches "done" or "abort" we can interrupt the watch process by "CTRL+c"

##### 4.3.4 let us define a directory for the results - for example the directory "output":

```
mkdir output
```

#### 4.3.5 let us gather the result:

```
glite-job-output --dir ./output -i job.list01
```

The output will be generated in under a generated directory composed by the user name and the long string returned by this command.

#### 4.3.6 Let us see what has happened (listings of standard output, and of standard error files)

```
cd output/<username>_<output of 4.3.5>
```

For example:

```
cd output/budapest13_yjca9Flm4gVY2AaqldKbw /  
ls -al  
cat std.out  
cat std.err
```

#### 4.3.7 Let us retrieve the grid file to our local system and compare the value of the file with the original.

```
lcg-cp --vo gilda  
sfn://grid005.iucc.ac.il/storage/gilda/generated/2006-06-  
28/filef78913b6-427a-4df0-806b-c7ea48ae0a04  
file:/home/budapest13/dum_Modified.
```