Draft Grid Storage Namespace Guidelines

J.J. Nebrensky (Brunel University, Uxbridge UB8 3PH, UK)

MICE-NOTE-COMP-247

The Grid can provide MICE not only with computing (number-crunching) power [1], but also with a secure global framework allowing users access to data. Although the focus is usually on the mass of experiment data, the Grid also opens up new possibilities for the storage and sharing of other material within the collaboration.

- Good news: storing development data on the Grid keeps it available to all MICE – not stuck on an old PC in the corner of the lab

- Bad news: loss of ownership – who picks up the data curation responsibilities?

This document provides an introduction to data storage on the Grid and describes the proposal for the directory structures to be used by MICE when registering data files stored on the Grid within a File Catalogue such as LFC.

Section 3 is based on the material presented at CM23, incorporating feedback received.

For more details about file storage on the Grid please refer to the gLite User Guide [2].

1 Data File Storage on the Grid

Interfaces to data storage on the Grid exist for both access to entire files and for querying databases, e.g. [3]; we are only concerned here with the former. The Grid itself is composed of a number of services that are accessed over existing Internet links, with a common security infrastructure based around digital certificates (PKI). Authentication and authorisation are based around the concept of a "*V*irtual *O*rganisation" (*VO*) which groups together users with a common interest and allows them access to shared resources.

Figure 1 illustrates file management on the Grid. Each file is referenced by a unique, machine-generated, global identifier, or *GUID*, when stored on the Grid. The file is physically uploaded to one (or more) *SE*s (*S*torage *E*lements) where it is known by a *SURL* (*S*torage *URL*) (possibly also machine-generated).

A service generically known as a "replica catalogue" tracks the multiple SURLs associated with a particular GUID. It should be noted that these are not merely copies of the original file, but explicitly identical replicas; if one is modified then it must be issued with a new GUID to preserve consistency.

Machine-generated names are not expected to be used by humans directly; instead it is possible to associate sensible filenames with each file (*LFN*, *L*ogical *F*ile *N*ame). A "file catalogue" is basically a database that maps between LFNs and the GUIDs needed to actually access the data on the Grid. MICE has a global instance of *LFC* (*L*CG *F*ile *C*atalogue, [4]) run by the Tier 1 centre at RAL. LFC can act as both replica and file catalogues, and allows LFNs to be organized in a directory tree with Unix-like access control.

For many applications – such as analysis – it is necessary to first select the list of files containing relevant data (rather than attempting to download and analyse every possible data file). The corresponding Grid service is known as a "metadata catalogue", as it searches through the metadata (essentially a set of keywords) associated with each file. AMGA [5] is an example of such a service. At the time of writing, no metadata catalogue has been set up for MICE yet.

If too many data transfers into/out of an SE are made simultaneously, then the network links will become congested and the server may become overloaded, causing some transfers to fail. The gLite *F*ile *T*ransfer *S*ervice (*FTS*, [6]) allows data movements to be requested in advance and will then schedule, carry out and validate the transfers while the user can be off-line.

The lower-level data handling in figure 1 may best be understood by considering the more complex form of SE as a generic example: one with both disk and tape storage facilities. This may consist of a number of separate *disk pool* servers (usually for short-term storage of data), a tape storage robot (for archival storage), and a *head node* that controls and co-ordinates access to the data. Let us use the example of downloading a file that happens to be stored on tape.

The first step is to ask for access to the file (specifying its SURL) from the SE head node. Communication with the head node usually uses the SRM protocol (hence a SURL will often start with `srm://`). The SE will then check that the file exists, and ask the tape robot to locate the tape on which the data is stored and insert it into a free tape drive. If the SE were simply to grant the client direct access to the tape drive, then users with low bandwidth connections would tie up the drive for a long time, blocking other transfers and reducing the efficiency of the tape system. Instead the SE will instead copy the file internally from the tape to a (dedicated) disk pool server. Once the file is ready for external transfer the SE replies to the client with a *TURL* (*T*ransfer *URL*) that specifies the endpoint from which the file may actually be transferred. This will usually be done using the GridFTP protocol (in effect, a traditional FTP transfer with Grid authentication/authorisation mechanisms added), hence TURLs will often begin with `gsiftp://`.

Thus, a **SURL** is a **long-term handle** by which a file is **requested**, while the ***TURL*** is an ***ephemeral location*** from which data is actually ***accessed***. This distinction is still relevant for disk-based SEs – although the TURL may appear to be persistent for a file stored on a disk pool server, routine maintenance operations often require data to be moved between filesystems or servers which will change the TURL.

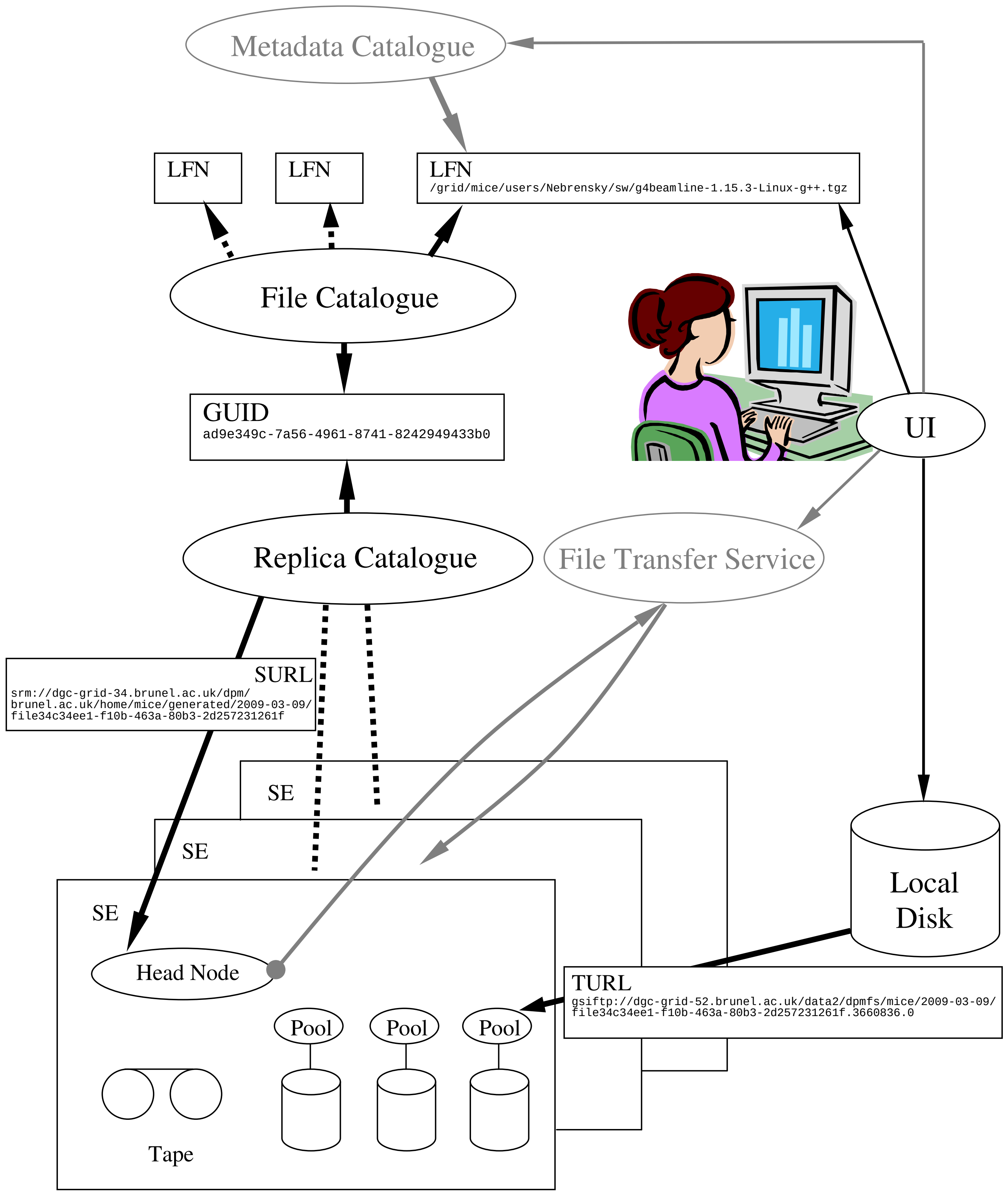
Metadata Catalogue
LFN
LFN
LFN
/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz
File Catalogue
GUID
ad9e349c-7a56-4961-8741-8242949433b0
UI
Replica Catalogue
File Transfer Service
SURL
srm://dgc-grid-34.brunel.ac.uk/dpm/
brunel.ac.uk/home/mice/generated/2009-03-09/
file34c34ee1-f10b-463a-80b3-2d257231261f
SE
SE
SE
Local Disk
Head Node
TURL
gsiftp://dgc-grid-52.brunel.ac.uk/data2/dpmfs/mice/2009-03-09/
file34c34ee1-f10b-463a-80b3-2d257231261f.3660836.0
Pool
Pool
Pool
Tape

Figure 1: Hierarchy of file storage services on the Grid (examples taken from section 2). Greyed out services have not yet been specified within MICE. Note that the user does not see the complexity of the Grid (as long as she doesn't turn round!).

MICE currently has access to one tape-based storage system – the CASTOR instance at the RAL Tier 1 centre. The other SEs supporting MICE are mostly disk-based systems running the “DPM” middleware [4].

Although figure 1 looks complex, users should not be dealing with each service individually; instead they should use the high-level middleware (this will also help to ensure data integrity). For example, to download a file from the Grid to local disk the user merely invokes the *lcg-cp* command: internally this will query the LFC to get the SURL; use SRM to get the corresponding TURL; and finally use GridFTP to fetch the data.

The security of the Grid is based on digital certificates that guarantee the identity of the end user. Access to resources (computing power and data) will only be granted to members of the MICE VO. The membership and mapping of certified identities to roles within a VO is done by a service called VOMS. The VOMS instance that serves the MICE VO is on the GridPP VOMS server [7], hosted at Manchester University (UK).

In order to help mitigate the consequences of their credentials being stolen, users do not give out their certificates directly but instead create short-lived proxies to represent them in their interactions with the middleware services. Thus, an SE will only allow access to data to those clients presenting unexpired proxies for which the VOMS server can confirm that the owner has suitable membership in the VO to satisfy any ACL restrictions applying to that file.

Note that during the lifetime of the experiment it is likely that some services will be replaced with other implementations. Although migration tools are expected to be available if needed, we should be wary of depending on specific foibles that may turn out to be implementation-specific.

Our current tools in MICE are based around the transfer of whole files to/from a local disk on the processing machine. The Grid also allows “POSIX I/O” (i.e. random access rather than sequential) directly to files on the local SE, typically using a secure version of the RFIO protocol. This would require compiling libshift into G4MICE, and would be useful only in cases where we need to access only a small part of the data in a file. We currently don't see any need for this in MICE.

2 Sample LFC Session

First, we create a proxy and set up the environment:

```
young:> voms-proxy-init -voms mice
Cannot find file or dir: /home/eesrjjn/.glite/vomses
Enter GRID pass phrase:
Your identity: /C=UK/O=eScience/OU=Brunel/L=ECE/CN=henry nebrensky
Creating temporary proxy ........................................................ Done
Contacting  voms.gridpp.ac.uk:15001
[/C=UK/O=eScience/OU=Manchester/L=HEP/CN=voms.gridpp.ac.uk/Email=ops@tier2.hep.manchester.ac.uk]
"mice" Done
Creating proxy .................................................................. Done
Your proxy is valid until Tue Mar 10 06:55:37 2009

young:> setenv LFC_HOST lfc.gridpp.rl.ac.uk
```

Next we check our LFC space and create a new directory, into which only we can write:

```
young:> lfc-ls /grid/mice/users/Nebrensky
LFCTest1r.for009.dat.gz
SETest.for009.dat.gz
fftw-3.0.1.tar.gz
t_test.gz
t_test2.gz

young:> lfc-mkdir -m 755 /grid/mice/users/Nebrensky/sw

young:> lfc-ls /grid/mice/users/Nebrensky/sw

young:>
```

Now we can upload a file (verbosely!) and add a comment to it:

```
young:> lcg-cr --vo mice --defaultsetype srmv2 -v -t 240 \
   -l "/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz" \
   "file://`pwd`/g4beamline-1.15.3-Linux-g++.tgz"
Using grid catalog type: lfc
Using grid catalog : lfc.gridpp.rl.ac.uk
Using LFN : /grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz
[BDII] lcg-bdii.gridpp.ac.uk:2170: Warning, no GlueVOInfo information found about SE 'dgc-
grid-34.brunel.ac.uk' (with no tag)
SE type: SRMv2
Using SURL : srm://dgc-grid-34.brunel.ac.uk/dpm/brunel.ac.uk/home/mice/generated/2009-03-
09/file34c34ee1-f10b-463a-80b3-2d257231261f
Alias registered in Catalog: lfn:/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-
g++.tgz
SRM Request Token: 44dff9d0-45a6-47c5-8363-44c7fb096c18
Source URL: file:/tmp/helloworld/mice/g4beamline-1.15.3-Linux-g++.tgz
File size: 29467308
VO name: mice
Destination specified: dgc-grid-34.brunel.ac.uk
Destination URL for copy: gsiftp://dgc-grid-52.brunel.ac.uk/dgc-grid-
52.brunel.ac.uk:/data2/dpmfs/mice/2009-03-09/file34c34ee1-f10b-463a-80b3-
2d257231261f.3660836.0
# streams: 1
# set timeout to 240 seconds
     24117248 bytes   8749.61 KB/sec avg  11264.00 KB/sec inst
Transfer took 4010 ms
Destination URL registered in Catalog:
srm://dgc-grid-34.brunel.ac.uk/dpm/brunel.ac.uk/home/mice/generated/2009-03-
09/file34c34ee1-f10b-463a-80b3-2d257231261f
guid:ad9e349c-7a56-4961-8741-8242949433b0

young:> lfc-setcomment /grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz \
   "Binary distro of g4beamline"

young:> lfc-ls  /grid/mice/users/Nebrensky/sw
g4beamline-1.15.3-Linux-g++.tgz

young:> lfc-ls -l --comment /grid/mice/users/Nebrensky/sw
-rw-rw-r-- 1 931 147 29467308 Mar 09 19:02 g4beamline-1.15.3-Linux-g++.tgz Binary distro
of g4beamline
```

We’re not publicly distributing g4beamline, so we’ll remove world access:

```
young:> lfc-chmod 640  /grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz

young:> lfc-ls -l /grid/mice/users/Nebrensky/sw
-rw-r----- 1 931 147 29467308 Mar 09 19:16 g4beamline-1.15.3-Linux-g++.tgz
```

Now we’ll pull the file back on to local disk, and check it’s OK:

```
young:> lcg-cp "lfn:/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz" \
   /tmp/g4bl.tgz
```

```
young:> diff -s g4beamline-1.15.3-Linux-g++.tgz /tmp/g4bl.tgz
Files g4beamline-1.15.3-Linux-g++.tgz and /tmp/g4bl.tgz are identical
```

Notice that at no point have we needed to know *where* on the Grid the file is stored – the system should choose a sensible default location when a file is first created, and after that LFC keeps track of the data. Out of interest we can check:

```
young:> lcg-lr lfn:/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz
srm://dgc-grid-34.brunel.ac.uk/dpm/brunel.ac.uk/home/mice/generated/2009-03-
09/file34c34ee1-f10b-463a-80b3-2d257231261f
```

Obviously, this is the SURL buried in the verbose output of the *lcg-cr -v* command above. For completeness, we'll now delete all replicas of the file from the Grid:

```
young:> lcg-del -a lfn:/grid/mice/users/Nebrensky/sw/g4beamline-1.15.3-Linux-g++.tgz

young:> lfc-ls /grid/mice/users/Nebrensky/sw

young:>
```

As we have finished, we destroy our proxy:

```
young:> voms-proxy-destroy

young:> lfc-ls -l /grid/mice/users/Nebrensky/sw
send2nsd: NS002 - send error : client_establish_context: Could not find or use a
credential
/grid/mice/users/Nebrensky/sw: Bad credentials
```

Without a valid proxy we cannot access LFC.

3 File Catalogue Namespace

This section is an updated version of the material presented at CM23 [8].

For uniformity and traceability, all data stored by MICE on the Grid should be registered in the file catalogue. LFC allows LFNs to be organized in a global shared directory tree with Unix-like user access control.

We need to agree on a consistent namespace for the file catalogue. The aim is NOT to replicate the experiment structure; rather we want an easy-to-understand structure that is compact but without too many entries in each low-level directory (aim for 20-50).

3.1 Principles

For many routine queries (e.g. analysis), the required filenames will actually be handled for us by the metadata catalogue service. However, in any situation where this is not possible then users will have to resort to a manual search – equivalents of *find*, *locate* and file-name completion don't yet exist. We can't assume it will be possible to browse this from a graphical interface with thumbnails – if you have to search through the LFC by hand, it will be painful even with the ideal structure as it will be a case of *ls* and eyeball…

We want to avoid deleting directories out of the LFC namespace - moving things will cause confusion (LFC allows multiple “soft” links, but this makes the whole structure bigger and more complex). Conversely, avoiding excess complexity should prevent creating dead branches in the first place.

MC simulations should be close to that which they model, rather than a separate branch from the very top.

Ideally a directory should contain either only more subdirectories, or only some files.

LFC is case-sensitive – use this to improve readability. Don’t even think of including spaces in filenames! **But**, need to be consistent in use of upper and lower case in LFNs, else will create bogus entries.

We need to bear in mind that information can be should be represented not only by the directory structures but also within the filenames themselves.

The Grid resources we will have access to will be driven by, and tuned for, LHC experiments. For best results we should aim for similar usage patterns (e.g. ~24 hour jobs, ~gigabyte files, etc.). [As an aside, note that for LFC – as for a number of other Grid services – the cryptographic authentication and authorisation steps can require more server resources than the transaction that they protect. Hence if a large number of queries is to be made it will be much more efficient to use the LFC API to carry out all the operations within a single session.]

3.2 Namespace (¾-baked proposal)

We get given `/grid/mice/` as the root by the server.

We propose four shared upper-level directories:

`MICE/`

> DAQ output and corresponding MC simulation. Split into MICE Step, and further divided e.g. by momentum, flip etc. (by Campaign for step 0). Separate directories for MC results, rather than encoding in filename. This stuff is the least likely to have people going through by hand. Example:
>
> `/grid/mice/MICE/Step`*`N/RefMomentum/`*`MC`
>
> This area of the namespace should be owned by some sort of “DAQ” or “Production” user or VOMS role.

`Construction/`

> historical data from detector development and QA. What about the structure relating to other modules? This material is unlikely to be accessed frequently –

should we bother with raw files, or just tarball/zip each subset up? Answers to both are probably obvious looking at each case in line with the concepts introduced above. Example:

```
/grid/mice
      /Construction/detector/data
```
e.g.
```
      /Construction/Tracker/StationQA
      /Construction/Tracker/OpticalQA
      /Construction/Tracker/FibreDamage
```

This area of the namespace will be owned by relevant individuals from each detector group.

`TestBeam/`

test beam data – KEK, Fermi, RAL (Tracker Cosmics?). The subdivisions to be place-specific, i.e.:

`/TestBeam/place/...`

I'm not sure who should own this tree!

`Calibration/`

large datasets needed during analysis. Example:

```
/grid/mice
      /Calibration/Tracker/VLPC
      /Calibration/BeamLine/FieldMaps
```

but e.g. are the solenoid field maps part of the spectrometer
```
      /grid/mice/Calibration/Spectrometer/FieldMaps
```
or part of the beamline
```
      /grid/mice/Calibration/BeamLine/FieldMaps/SpectrometerSolenoid
```
or do we put the field maps together
```
      /grid/mice/Calibration/FieldMaps/SpectrometerSolenoid
      /grid/mice/Calibration/FieldMaps/Quads
```

It's still not clear how should this be split up – deciding this requires knowing what will be in here as opposed to the 'mythical' DB.

This area of the namespace should probably be owned by the detector groups providing the data.

As we are starting to really use LFC now, we need at least a crude idea of

- what calibration data will be need to be stored on the Grid

- how it will be provided - O(# of files), format, size

I have no (= zero) idea of what to expect from most of MICE for this !!!

For completeness, note that there are two other upper-level directories already in place (note capitalisation):

`generated/`

> This is automatically created by the catalogue to track things with no name.

`users/`*`name`*

> This provides an area for people to use as scratch space for their own purposes such as simulation output. We encourage people to do this through LFC rather than lower level mechanisms because it's easier, and also as this helps avoid "dark data" – files taking up space on disk that the VO management doesn't know about, and can't easily delete to make more room (or, indeed, request sites **not** to delete, to make more room!). N.B. again - LFC allows Unix-style access permissions.

4 Comments ("Metadata") Field

As has been seen in the samples session (section 2), LFC allows a single optional comment field for each file entry (although this is sometimes referred to as a metadata field, it does not appear to be possible to base a search on it). Two possible standardised uses for this field have been proposed within MICE:

- *checksum*: The field could be set to e.g. the MD5 hash of the data file, to help detect data corruption. In practice however we expect that data files will be compressed e.g. with *gzip*, and since they must be downloaded anyway to calculate the confirmatory checksum there is no advantage over simply testing the archive itself, e.g. *gunzip –t*.

- *file format*: The field could contain details about the format in which the data is stored, such as any compression applied (*gzip* vs. *zip*). For such single-item descriptions this could simply use MIME-style entries [9]; however we may need to extend this or find some other scheme. For example, an ASCII text file may be an input to Turtle (rather than g4Beamline), but it may also be either an *input deck* (beamline description) or input *beam* (ensemble of muons); and it may be compressed for storage by one of several methods (*gzip*, *zip*, or – of course – *none*); thus requiring three descriptors.

Although the latter option looks more promising, it may need a lot more work to be useful.

5 Related Environment Variables

For consistency across MICE activities, I would also propose two related environment variables:

MICE_GRID_PWD represents the current "working directory" within the LFC namespace; for the majority of the sample session above this would be "`/grid/mice/users/Nebrensky/sw`". Thus to download a file the source argument to *lcg-cp* will be "`lfn:${MICE_GRID_PWD}/filename`".

MICE_HOME_SEID contains the name of a Storage Element that a Grid user considers to be local to themselves and which they know works. Generally, when new data files are created by a Grid job at a remote cluster they should be stored at an SE close to that cluster – usually on the same fast LAN segment – to free up the compute nodes as soon as possible. As remote SEs may occasionally have problems even when the rest of the site is OK, MICE_HOME_SEID will allow Grid job wrapper scripts to fall back to storing the data at the user's SE if the transfer to the site SE fails. As an example, this transfer failover mechanism has been implemented in the latest version of the g4beamline wrapper script [10].

These names do not clash with existing Grid practice [11].

6 Conclusion

At the time of writing many details of the LFC namespace are still unclear and would benefit from stakeholder input. Speak now or forever hold your peace!

A revised version of this Note will be circulated in due course.

References

1. D. Forrest: "*The Grid & MICE*" MICE Note 246 (2009)

2. S. Burke, S. Campana, E. Lanciotti, P. Méndez Lorenzo, V. Miccio, C. Nater, R. Santinelli and A. Sciabà: "*gLite 3.1 User Guide*" CERN-LCG-GDEIS-722398 v. 1.2, 7th January 2009. https://edms.cern.ch/document/722398

3. OGSA-DAI http://www.ogsadai.org.uk/

4. "*Official Documentation for LFC and DPM*" https://twiki.cern.ch/twiki/bin/view/LCG/DataManagementDocumentation

5. B. Koblitz and N. Santos: *"AMGA User's and Administrator's Manual"* v. 1.9.0 29th October 2008 http://amga.web.cern.ch/amga/downloads/amga-manual-1.9.0.pdf

6. FTS: see https://twiki.cern.ch/twiki/bin/view/LCG/FtsWlcg or https://twiki.cern.ch/twiki/bin/view/EGEE/FTS

7. GridPP http://www.gridpp.ac.uk/

8. H. Nebrensky: "*Grid Update*" MICE Collaboration Meeting (CM23), January 2009

9. "*Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*" IETF/NWG RFC 2046 http://www.ietf.org/rfc/rfc2046.txt (1996)

10. H. Nebrensky: "*Various bits of Grid-related MICE stuff*" http://people.brunel.ac.uk/~eesrjjn/mice/mice_grid.htm

11. "*Grid Batch Job Environment Variables*" EGEE-SA1-TEC-GridBatchJobEnvironmentVariable-v2.1.doc, 29th August 2005 https://edms.cern.ch/document/630962