



LHC Computing Grid

MANUALS SERIES LCG-1 USER GUIDE

Document identifier: **LCG-xxxx**
Date: **13/10/2003**
Authors: **Flavia Donno, Andrea
Sciabà**
Document status: **DRAFT**

Document Change Record

| Issue | Item | Reason for Change |
|----------|------|--|
| 16/06/03 | 0_0 | First Draft – Introduction: Flavia Donno, Job Management: Andrea Sciaba’ |
| 23/06/03 | 0_1 | Data Management Additions: Flavia Donno |
| 22/07/03 | 0_2 | Checking all, adding release details and getting started details: Flavia Donno |
| 24/07/03 | 0_3 | Add services interaction: Flavia Donno |
| 26/07/03 | 0_4 | Merge with Andrea’s changes on certificates: Andrea Sciaba’ |
| 15/08/03 | 0_5 | Insert some details about the Information System: Flavia Donno |
| 18/08/03 | 0_6 | More details about GLUE Schema: Flavia Donno |
| 20/08/03 | 0_7 | More details about Data Management Tools: Flavia Donno |
| 27/08/03 | 0_8 | Appendix on GLUE schema and details about Job Management: Andrea Sciaba’ |
| 31/08/03 | 0_9 | Site GIIS and BDII: Flavia Donno |
| 01/09/03 | 0_10 | Additions to the appendix on the GLUE schema |
| 10/09/03 | 0_11 | Corrections from comments by Ian Neilson and Maria Dimou |
| 26/09/03 | 0_12 | Added section on BrokerInfo and expanded the appendix on Glue schema |
| 2/10/03 | 0_13 | Added PEM to P12 conversion, added example on changing default VO, added requirements in JDL, added appendix on job states |

Files

| Software Products | User files |
|-------------------|---------------------|
| Word | LCG-1_UserGuide.doc |
| | |

Content

| | | |
|-----------|---|-----------|
| 1 | INTRODUCTION | 5 |
| 1.1 | OBJECTIVES OF THIS DOCUMENT | 5 |
| 1.2 | APPLICATION AREA | 5 |
| 1.3 | DOCUMENT EVOLUTION PROCEDURE | 5 |
| 1.4 | APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS | 6 |
| 1.5 | TERMINOLOGY | 8 |
| 2 | EXECUTIVE SUMMARY | 9 |
| 3 | OVERVIEW: THE LCG-1 SERVICE | 10 |
| 3.1 | THE LCG-1 ARCHITECTURE | 11 |
| 3.2 | SERVICE INTERACTIONS AND JOB FLOW | 16 |
| 3.2.1 | <i>Job submission</i> | 16 |
| 3.2.2 | <i>Data Management</i> | 18 |
| 3.2.3 | <i>Information System</i> | 19 |
| 4 | GETTING STARTED | 21 |
| 4.1 | REGISTERING WITH LCG | 22 |
| 4.2 | OBTAINING A CERTIFICATE | 21 |
| 4.3 | VIRTUAL ORGANIZATIONS | 22 |
| 4.4 | SETTING UP THE USER ACCOUNT | 23 |
| 4.5 | CHECKING A CERTIFICATE | 23 |
| 4.6 | PROXY CERTIFICATES | 25 |
| 4.7 | ADVANCED PROXY MANAGEMENT | 27 |
| 5 | JOB MANAGEMENT | 29 |
| 5.1 | THE COMMAND LINE INTERFACE | 29 |
| 5.1.1 | <i>Job submission</i> | 29 |
| 5.1.2 | <i>Job description language (JDL)</i> | 31 |
| 5.1.3 | <i>Checkpointable jobs</i> | 34 |
| 5.1.4 | <i>Interactive jobs</i> | 34 |
| 5.1.5 | <i>Job operations</i> | 34 |
| 6 | DATA MANAGEMENT | 38 |
| 6.1 | THE EDG-REPLICA-MANAGER CLIENT TOOLS | 39 |
| 6.2 | THE EDG-LOCAL-REPLICA-CATALOG AND EDG-REPLICA-METADATA-CATALOG CLIENT TOOLS | 44 |
| 7 | INFORMATION SYSTEM | 48 |
| 7.1 | THE LOCAL GRIS | 48 |
| 7.2 | THE SITE GIIS | 51 |
| 7.3 | THE BDII | 54 |
| 8 | APPENDIX A | 55 |
| 8.1 | THE GRID MIDDLEWARE | 55 |
| 9 | APPENDIX B | 57 |
| 9.1 | THE GLUE SCHEMA | 57 |
| 9.1.1 | <i>Attributes for the Computing Element</i> | 57 |
| 10 | APPENDIX C | 65 |
| 10.1 | TROUBLESHOOTING | 65 |



| | | |
|--------|--------------------------|----|
| 10.1.1 | Job Submission | 65 |
| 10.1.2 | Data Management..... | 65 |
| 10.1.3 | Information System | 65 |

1. INTRODUCTION

1.1. OBJECTIVES OF THIS DOCUMENT

1.2. APPLICATION AREA

1.3. DOCUMENT EVOLUTION PROCEDURE

This document will be modified accordingly with the LCG-1 releases to reflect always the status of the LCG-1 service.

1.4. APPLICABLE DOCUMENTS AND REFERENCE DOCUMENTS

Applicable documents

| | |
|------|--|
| [A1] | |
| | |
| | |

Reference documents

| | |
|------|--|
| [R1] | Regional Centres for LHC computing The MONARC Architecture Group http://barone.home.cern.ch/barone/monarc/RCArchitecture.html http://monarc.web.cern.ch/MONARC/ |
| [R2] | The Anatomy of the Grid. Enabling Scalable Virtual Organizations Ian Foster, Carl Kesselman, Steven Tuecke http://www.globus.org/research/papers/anatomy.pdf |
| [R3] | Overview of the Grid Security Infrastructure http://www-unix.globus.org/security/overview.html |
| [R4] | Resource Management http://www-unix.globus.org/developer/resource-management.html |
| [R5] | The GridFTP Protocol and Software http://www.globus.org/datagrid/gridftp.html |
| [R6] | MDS 2.2 Features in the Globus Toolkit 2.2 Release http://www.globus.org/mds/ |
| [R7] | The GLUE CE schema http://www.cnaf.infn.it/~sergio/datatag/glue/v11/CE/index.htm http://www.cnaf.infn.it/~sergio/datatag/glue/index.htm |
| [R8] | Job Description language HowTo http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.pdf |
| [R9] | User Guide for EDG Replica Manager 1.4.8 http://proj-grid-data-build.web.cern.ch/proj-grid-data-build/edg-replica-manager/user-guide/html |

| | |
|-------|--|
| [R10] | User Guide for EDG Replica Optimization 2.0.2 http://proj-grid-data-build.web.cern.ch/proj-grid-data-build/edg-ros/user-guide/html |
| [R11] | EDG Local Replica Catalog 2.0.2 http://proj-grid-data-build.web.cern.ch/proj-grid-data-build/edg-rls-server/user-guide/html |
| [R12] | EDG Replica Metadata Catalog 2.0.2 http://proj-grid-data-build.web.cern.ch/proj-grid-data-build/edg-metadata-catalog/user-guide/html |
| [R13] | EDG Tutorial – Handout for Participants for EDG Release 2.x http://edms.cern.ch/document/393671 |
| [R14] | WP1 Workload Management Software – Administrator and User Guide Sep 4th, 2003 http://server11.infn.it/workload-grid/documents.html |

1.5. TERMINOLOGY

Definitions

| | |
|--|--|
| | |
| | |
| | |

Glossary

| | |
|--|--|
| | |
| | |
| | |

2. EXECUTIVE SUMMARY

This user guide is intended for users of the LCG-1 service. In this guide, the user will hopefully find an adequate introduction to the services provided and a description on how to use them. Examples are given for Job Management, Data Management, Resource Status, Monitoring, etc. in order to easily be effective. Troubleshooting hints are also available.

After a first introduction on the organization of the service itself presented in Chapter ..., an overview of the Workload Management service is given.....

....

3. OVERVIEW: THE LCG-1 SERVICE

The job of the LHC computing Grid Project – LCG – is to prepare the computing infrastructure for the simulation, processing and analysis of LHC data for all four of the LHC collaborations: **ALICE**, **ATLAS**, **CMS** and **LHCb**. This includes both the common infrastructure of libraries, tools and frameworks required to support the physics application software, and the development and deployment of the computing services needed to store and process data, providing batch and interactive facilities for the worldwide community of physics involved in LHC.

The requirements for LHC data handling are very large, in terms of computational power, data storage capacity, data access performance and the associated human resources for operation and support. It is not considered feasible to fund all of the resources at one site, and so it has been agreed that the LCG computing service will be implemented as a geographically distributed Computational Data Grid. This means that the service will use computing resources, both computational and storage, installed at a large number of Regional Computing Centres in many different countries, interconnected by fast networks. Special software, referred to generically as **Grid Middleware**, will hide much of the complexity of this environment from the user, giving the impression that all of these resources are available in a coherent virtual computer centre.

In the first phase of the project, from 2002 through 2005, LCG will develop and prototype the computing services and the operation of a series of computing data challenges of increasing size and complexity to demonstrate the effectiveness of the software and computing models selected by the experiments.

The LCG-1 service is the first data intensive production Grid. Following the Monarc model [R1] the first sites participating to this Grid are Tier 1 centres: BNL, CERN, CNAF, FNAL, FZK, IN2P3, MOSCOW, RAL, TAIWAN, TOKYO, with NIKHEF providing some authorization services.

LCG-1 is organized into Virtual Organizations [R2]: dynamic collections of individuals and institutions sharing resources in a flexible, secure and coordinated manner. In such settings, we encounter unique authentication, authorization, resource access, resource discovery, and other challenges.

3.1. THE LCG-1 ARCHITECTURE

In this section we give an overview of the LCG-1 architecture.

Before LCG resources can be used, a user is required to register some personal data and information about the Virtual Organization he/she belongs to with the **LCG Registration Server**. CERN will run such a service, collecting information about all LCG users.

The Grid Security Infrastructure (GSI) in LCG-1 enables secure authentication and communication over an open network [R3]. GSI is based on public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) communication protocol. Extensions to these standards have been added for single sign-on and delegation.

In order to access Grid resources, a user needs to have a digital X509 certificate from a **Certification Authority** (CA) recognized by LCG. The CA's recognized by LCG are listed later on.

There are five possible Virtual Organizations (VOs) a user can be affiliated to. A **Virtual Organization Server** maps users certificates to users data and lists the certificates as belonging to users part of a VO or LCG HEP experiment. The VO Server for the *DTeam* (LCG Grid Deployment Group) is run at CERN, while the VO Servers for the four HEP experiments are run at NIKHEF.

A user is authorized to use LCG-1 Grid resources by means of the **grid-mapfile** mechanism. Each host, part of the LCG-1 Grid, has a local grid-mapfile which maps user certificates to local account. When a user request-for-service reaches a host, the certificate of the user is checked in the local grid-mapfile. If the user certificate is found there, then the local account to which the user certificate is mapped is used to serve the request. The same is true for services. Details are explained in [R3].

The following paragraphs describe several types of services run in LCG-1 to provide the Grid functionality.

The initial point of access to the LCG-1 Grid is the **User Interface (UI)**. This is a machine where LCG users have a personal account and where user's certificate is installed. This is the gateway to Grid services. From the UI a user can be authenticated and authorized to use the LCG-1 Grid resources. This is the component that allows users to access the functionalities offered by the workload, data and information management systems. It provides a command user interface to perform some basic grid operations:

- submit a job for execution on a computing element;
- list all the resources suitable to execute a given job;
- replicate and copy files
- cancel one or more jobs;

-
- retrieve the output of one or more finished jobs;
 - show the status of one or more submitted jobs.

Normally one or more UI's are available at each site part of the LCG-1 Grid.

A **Computing Element (CE)** is defined as a Grid batch queue and it is identified by a pair "hostname/batch queue name". A Computing Element is a homogeneous farm of computing nodes called **Worker Nodes (WN)** and a node acting as a **Grid Gate (GG)** or front-end to the rest of the Grid. The GG runs a Globus gatekeeper, the Globus GRAM (Globus Resource Allocation Manager) [R4], the master server of a batch system (such as PBS) together with EDG Logging and Bookkeeping Services [R14]. In LCG-1 the batch systems supported are PBS, LSF and Condor. While all WNs can be hidden and running behind a firewall, the Gate node must be accessible from outside the site. The GG is responsible for accepting jobs and dispatching them for execution to the WNs. The GG provides a uniform interface to the computational resources it manages. On the WNs all commands and API for performing actions on Grid resources and Grid data are available.

Each LCG-1 site runs at least one CE and a farm of WNs behind it.

A **Storage Element (SE)** provides uniform access and services to large storage spaces. The storage element may control large disk arrays, mass storage systems and the like. The LCG-1 Grid provides support only for disk storage and not yet for robotic tape library devices. A GridFTP server [R5] runs on the Storage Element. It is responsible for secure, fast and efficient file transfer to/from the Storage Element.

Each LCG-1 site provides one or more SEs.

The resources described up to now constitute the compute and storage power of the LCG-1 Grid.

Together with the infrastructure described above, additional services are provided to locate and report on the status of Grid resources, to find the most appropriate resources to run a job requiring certain data access and to automatically perform data operations necessary before and after a job is run.

In the first release of the LCG-1 service the **Monitoring and Discovery Service (MDS)** from Globus [R6] has been adopted. This service provides information about the LCG-1 Grid resources and their status.

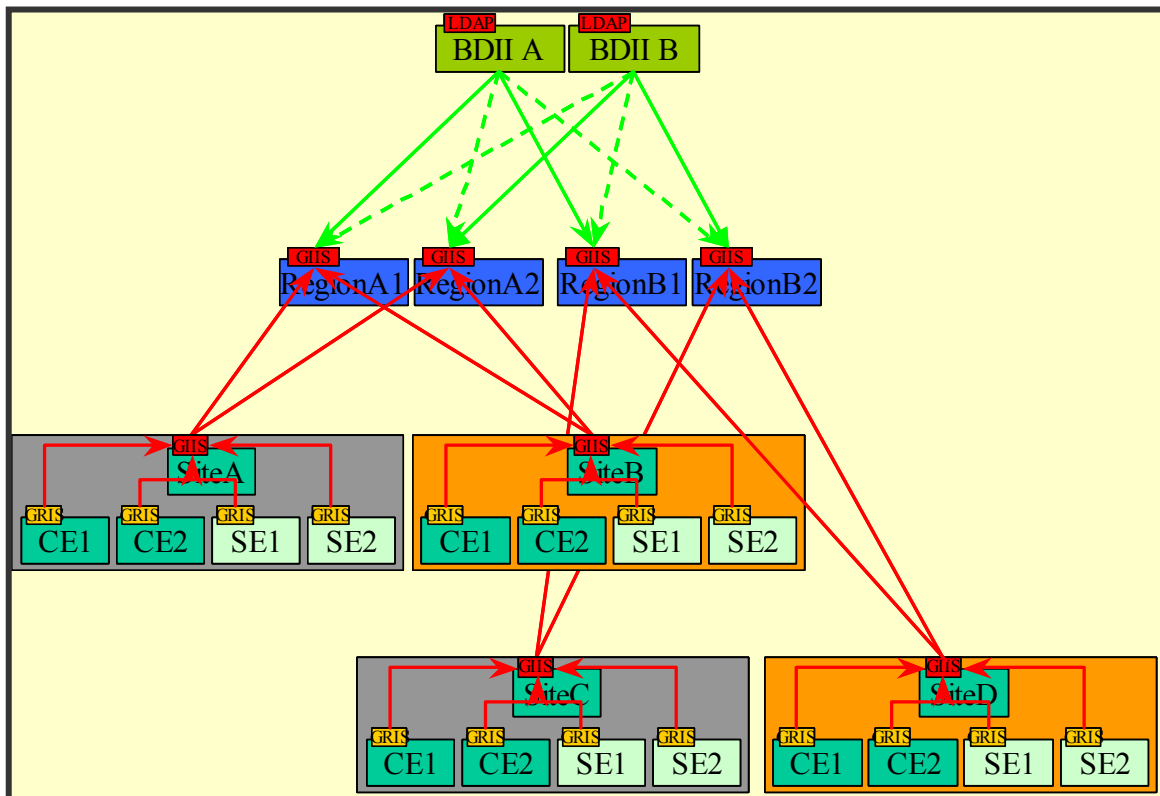


Figure 1

Figure 1 shows how the information is stored and propagated. Information is propagated in a hierarchy: Compute and Storage resources at a site report (via the **Grid Resource Information Servers, or GRISes**) their static and dynamic status to the **Site Grid Index Information Server (GIIS)**. In LCG-1, the site GIISes register with one or more Regional GIISes, for redundancy reasons and to nicely divide the administration domains. Figure 2 shows the sites hosting **Regional GIISes**. This structure is transparent to the user.

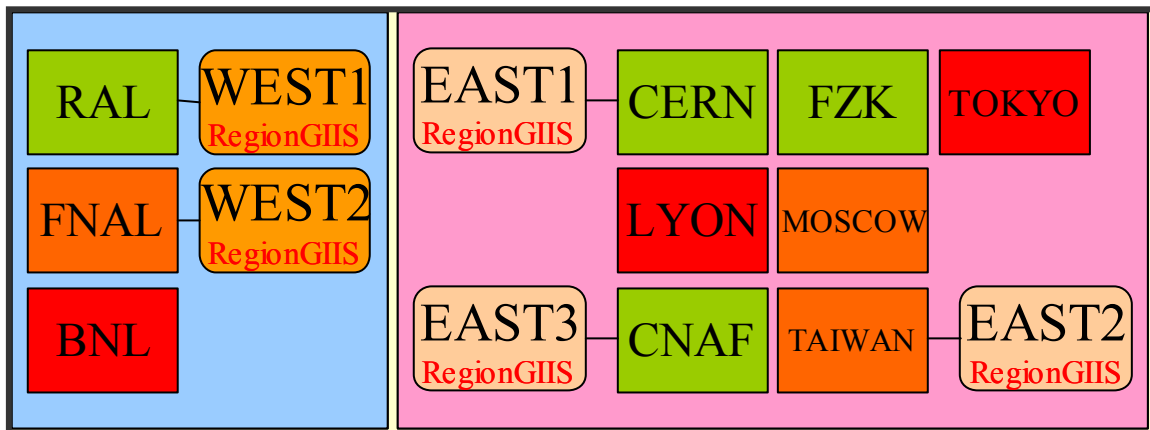


Figure 2

Due to dynamic nature of the GRID, the GIISes might not contain information about resources that are actually available on the Grid but that, for some reasons, are unable to publish to the GIISes updated information. Because of this, the **Berkeley DB Information Index (BDII)** was introduced. The BDII queries the regional GIISes and acts as a cache storing information about the Grid status in its database. Every time a resource appears in one of the GIISes, its existence is registered in one of the BDIIs. There is one BDII running at a site where a Resource Broker (RB, see later) is installed. Users and other Grid services (such as the RB) can interrogate BDIIs to get information about the Grid status. Very up-to-date information can be found by directly interrogating the site GIISes or the local GRISes that run on the specific resources. Later on we describe how a user can interrogate these services.

Information about data location and metadata entries can be found in the **Replica Location Service (RLS)**. In order to improve data access on the Grid, data files are replicated at different sites. This is done using the Data Management Services provided by the European DataGrid (EDG). In the LCG-1 Grid, files are uniquely identified by the Grid Universal ID (GuID). In order to keep track of data location, two catalogues are provided: the **Local Replica Catalog (LRC)**, which stores the mapping between the GuID associated with a file and its physical locations at a given site; and the **Replica Metadata Catalogue (RMC)** which stores the mapping between GuIDs and filenames or Aliases assigned by users (see *Figure 3*).

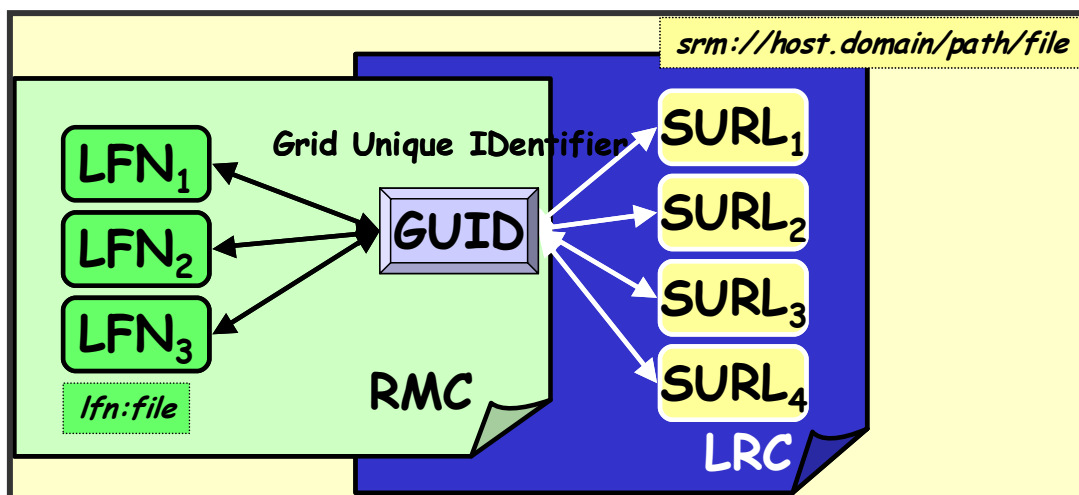


Figure 3

Consistency between RMC and LRC catalogues is unfortunately not always guaranteed.

For the moment these catalogues are centralized and there is one RLS per VO. In the first phase, all RLS are run at CERN.

The **Resource Broker (RB)** is the machine where the services of the **Workload Management System**, or **WMS** (Network Server, Workload Manager with the Match-Maker and Job Adapter Helpers, and Job Control Server) run. It is responsible for accepting job requests from users, interrogating the Information Index Servers (BDII) and the RLS, matching job requirements to available resources at various sites within LCG-1 Grid and passing jobs for dispatch to the **Job Control Service (JCS)**.

The **Logging and Bookkeeping Service (LBS)** logs all job management Grid events which can then be retrieved by users or system administrators for monitoring or troubleshooting. The LBS is usually run on the same machine where the RB runs.

Multiple RBs are available in LCG-1 Grid. Participating sites are free to install their own RBs. To know which sites have installed an RB, the LCG-1 deployment status page can be consulted at:

<http://cern.ch/grid-deployment/cgi-bin/index.cgi?var=lcg1Status>

The last component of the LCG-1 Grid described here is the **Proxy Server (PS)**. When a user accesses the Grid, he/she is provided with a temporary certificate, called proxy, that has an expiration time. If the user proxy expires before the

user job has finished, all subsequent requests for service will fail due to unauthorized access. In order to avoid this, the Workload Management Service provided by EDG allows for proxy renewal before the expiration time has been reached if the job requires it. The PS is the component that allows for such functionality.

In LCG-1 a site is free to install a PS. To know which sites have installed a PS, the LCG-1 deployment status page can be consulted at:

<http://cern.ch/grid-deployment/cgi-bin/index.cgi?var=lcg1Status>

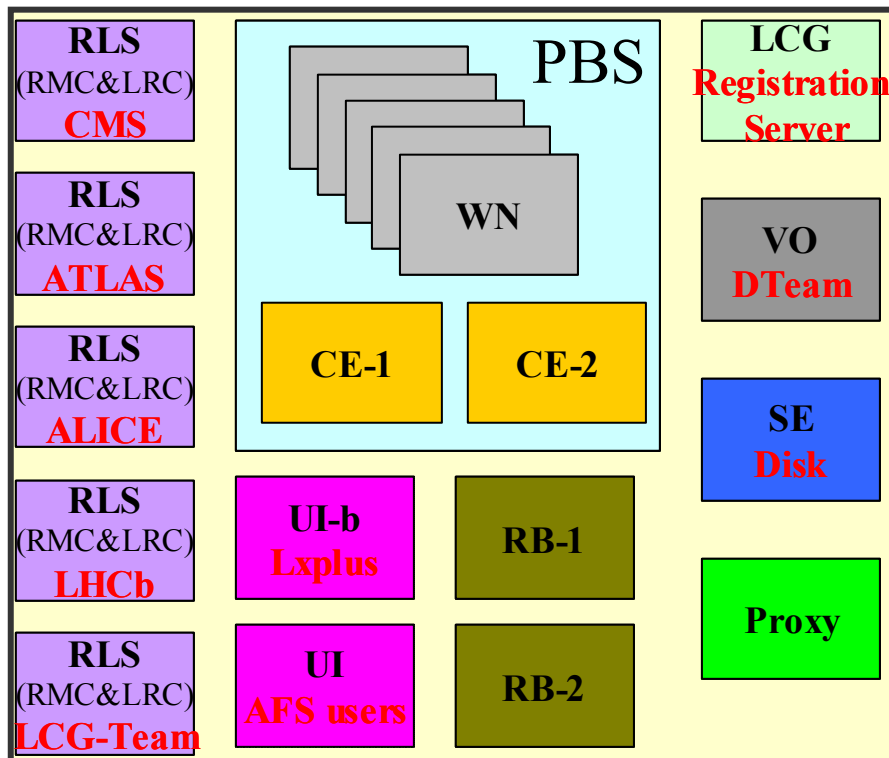


Figure 4

Figure 4 shows a summary of all LCG-1 service components available at CERN.

3.2. SERVICE INTERACTIONS AND JOB FLOW

In what follows we describe briefly what happens when a user submits a job to the LCG-1 Grid to process some data and how the different components interact. We also give a description of the components of the Data Management System. User applications and further functionality can be built/developed on top of what is offered by LCG-1 Grid.

3.2.1. Job submission

-
- a. After obtaining a digital certificate from one of the LCG-1 trusted Certification Authorities, registering with LCG, registering with a Virtual Organization and obtaining an account on an LCG-1 User Interface (once only actions), the user is ready to use LCG-1 Grid. He/she logs to the UI machine and creates a proxy certificate that authenticates him/her in every secure interaction, and has a limited lifetime.
 - b. The user submits the job from the UI to the Workload Management System (WMS), where a job is intended as a program to be executed on a computing node. The user can specify in the job description file one or more files to be copied from the UI to the RB node; this set of files is called "input sandbox" . The event is logged in the LB and the status of the job is **SUBMITTED**.
 - c. The WMS, and in particular the Match-Maker component, performs the matchmaking to find the best available CE to execute the job. To do so, the Match-Maker interrogates the BDII to query the status of Computational and Storage Resources and the RLS to find location of data. The event is logged in the LB and the status of the job is **WAIT**.
 - d. The WMS Job Adapter prepares the job for submission creating a wrapper script that is passed, together with other parameters, to the Job Controller for submission to the selected CE. The event is logged in the LB and the status of the job is **READY**.
 - e. The Globus Gatekeeper on the CE receives the request and sends the Job for execution to the Local Resource Management System (LRMS), such as PBS, LSF or Condor. The event is logged in the LB and the status of the job is **SCHEDULED**.
 - f. The LRMS handles the job execution on the available local farm worker nodes. User's files are copied from the RB to the WN where the job is executed. The event is logged in the LB and the status of the job is **RUNNING**.
 - g. While the job runs, Grid files can be accessed on the (close) SE using either the rfio protocol or POSIX I/O if the close SE serves its filesystems to the WN. In order for the job to find out which is the close SE, or what is the result of the Match-Maker process, a file with this information is produced by the WMS and shipped together with the job to the WN. This is known as the .BrokerInfo file. Information can be retrieved from this file using the BrokerInfo Command Line Interface (CLI) or the Application Programming Interface (API) library.
 - h. The job can produce new output data that can be uploaded to the Grid and made available for other Grid users to use. This can be achieved using the Data Management tools described later. Uploading a file to the Grid means to copy it on a Storage Element registering its location, metadata and attribute to the RLS. At the same time, during job execution or from the User Interface, data files can be replicated between two SEs using again the Data Management tools.

- i. When the job reaches the end without errors, the output (not large data files, but just small output files specified by the user in the so called "output sandbox") is transferred back to the RB node. The event is logged in the LB and the status of the job is **DONE**.
- j. At this point, the user can retrieve the output of his/her job from the UI using the WMS CLI or API. The event is logged in the LB and the status of the job is **CLEARED**.
- k. Queries of the job status are addressed to the LB database from the UI machine. Also, from the UI is it possible to query the BDII for a status of the resources.

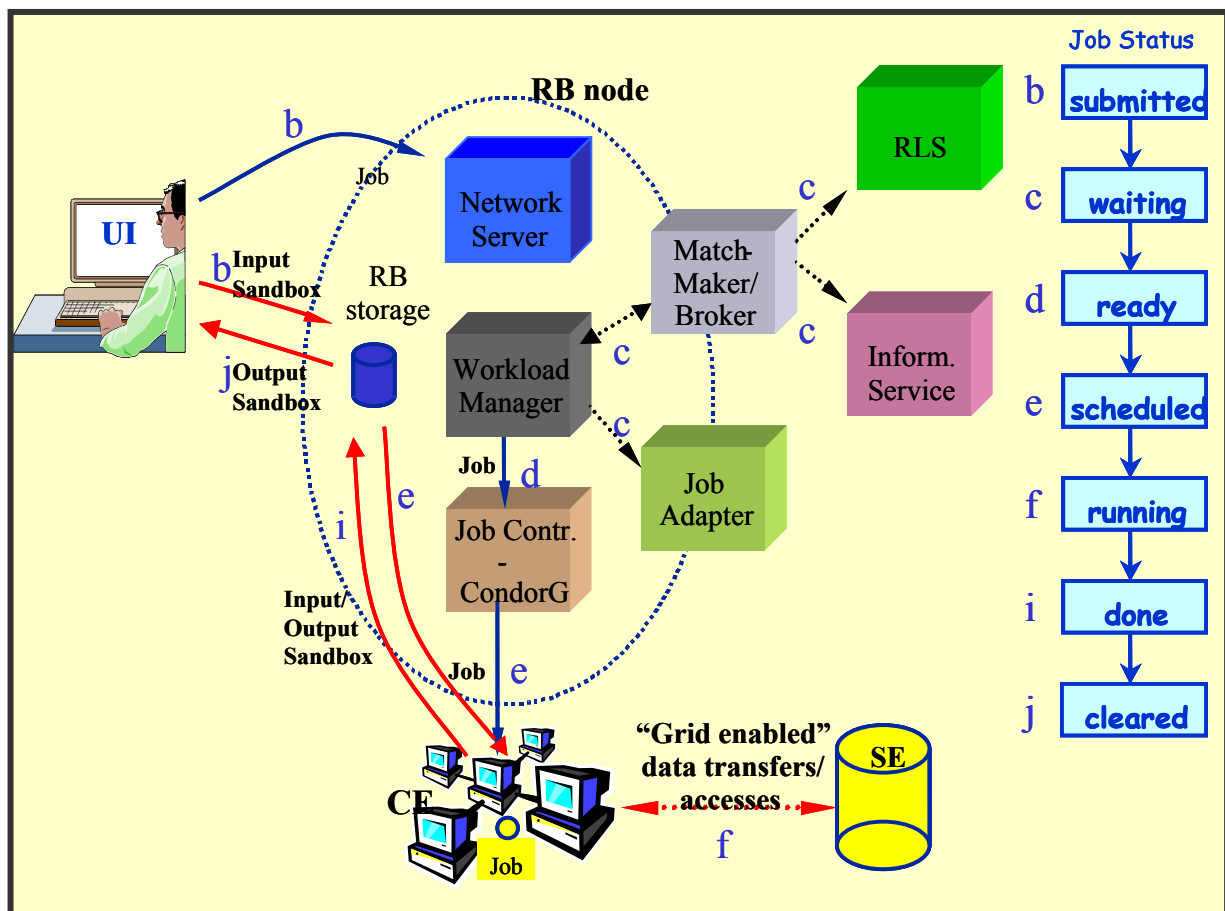


Figure 5

Figure 5 shows what has been described in steps b to k.

3.2.2. Data Management

The Input/Output Sandbox is a mechanism for transferring over the Grid small data files needed to start the job or to check the final status. Large data files are available on the Grid and known to other users only if they are stored on SEs and registered in the RLS catalogues. In order to optimise data access and to introduce fault-tolerance and redundancy, data files can be replicated on the Grid. The EDG Replica Manager, the Replica Location Service and the Replica Metadata Service are the tools available for performing these tasks. Only anonymous access to the Data Catalogues is supported: the user proxy is not used to control access to them.

On the LCG-1 Grid, a file is identified by the Grid Universal Identifier (GUID), a character string randomly generated by the EDG Replica Manager that uniquely identifies the file. However, a user normally refers to a file via a meaningful name or alias. Also, the same file might be replicated on different SEs on the Grid with different physical full filenames or even stored on a Mass Storage System with a tape library backend. Ownership of the file, its attributes such as checksum or size, and other metadata information need to be stored as well. The EDG Data Management system relies on two catalogue services: the Local Replica Catalog storing the association between GUIDs and physical location of files **at a site**, together with file attributes; and the Replica Metadata Catalog storing the association between the user filenames or aliases of a file to the file GUID. The user should never directly interact with these two catalogues, but always through the EDG Replica Manager.

- l. When a new file is produced, the file should be uploaded to the Grid to be known and usable by Grid services or other Grid users. This can be done using the EDG Replica Manager commands for copying and registering a file.
- m. Before running a job on the Grid, the user can ask the WMS to run the job on a CE close to an SE containing the data of interest, or, at run time, the job can ask the Replica Management Services to replicate a file on a SE close or even on the WN where the job is running.
- n. If a file is no longer needed, it can be deleted from the Grid and all its references removed from the Data Catalogues.

3.2.3. Information System

We have already illustrated the architecture of the Information System in LCG-1 Grid. Users can interrogate the Information System to retrieve static or dynamic information about the status of the LCG-1 Grid. In order to have an optimal answer, users are encouraged to query the BDII or the Site GIISes, but not the regional GIISes. Also, the specific GRISes can be queried. Details and examples on how to interrogate GRIS, GIIS and BDII are given later.

The Information System is based on OpenLDAP. LDAP is a protocol that provides the infrastructure for a directory service. A directory service is a specialized database optimized for reading, browsing and searching information. No transaction or roll-back features are normally offered. In particular in LCG-1 Grid, only anonymous access to the catalogue is offered. **This means that all users**

can browse the catalogues and all services are allowed to enter information into it.

The LDAP information model is based on entries.

An *entry* is a collection of *attributes* which together form a globally unique Distinguished Name (DN), a name that uniquely identifies the entry.

Each of the entry's attributes has a type and one or more *values*. The types are typically mnemonic strings, like "cn" while the syntax of the values depends on the attribute type. An LDAP *schema* describes the attributes and the types of the attributes associated with entries.

Directory entries are arranged in a hierarchical tree-like structure referred to as Directory Information Tree (DIT) as shown in *Figure 6*.

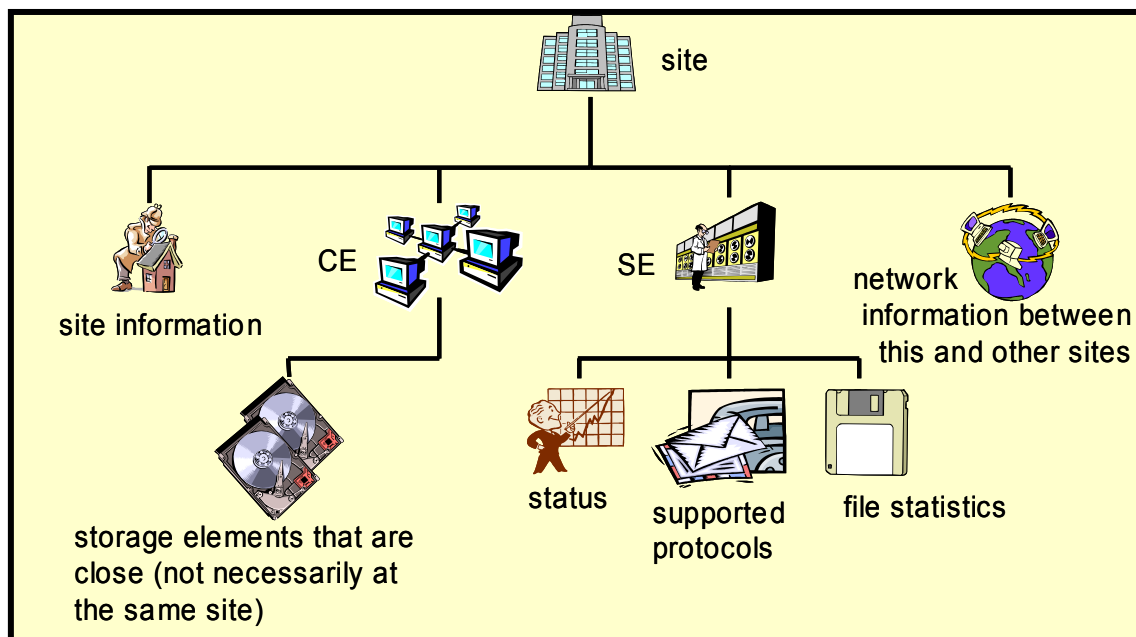


Figure 6

LCG-1 Grid deploys the GLUE (Grid Laboratory for a Uniform Environment) Schema for information description. The GLUE Schema activity aims to define a common conceptual data model to be used for grid resources monitoring and discovery. There are four main components of the GLUE Schema. They describe the attributes and value of Computing Elements, Storage Elements, binding information for Computing and Storage Elements, and Network Elements. Details can be found in [R7]. Examples on how to query the Information System in LCG-1 are given later on.

4. GETTING STARTED

This section describes the preliminary steps to gain access to the LCG-1 Grid. Before using the LCG-1 Grid, the user must do the following:

1. Obtain a cryptographic X.509 certificate from an LCG-1 approved Certification Authority (CA).
2. Get registered with LCG.
3. Join one of the LCG-1 Virtual Organizations (VOs).
4. Obtain an account on a machine which has the LCG-1 User Interface software installed.
5. Create a proxy certificate.

Steps 1 to 4 need to be executed only once to have access to the Grid. Step 5 needs to be executed the first time a request to the Grid is submitted. It generates a proxy valid for a certain period of time. At the proxy expiration, a new proxy must be created before the Grid services can be used again.

The following sections provide details of the prerequisites.

4.1. OBTAINING A CERTIFICATE

The first requirement the user must fulfil is to be in possession of a valid X.509 certificate issued by a recognized certification authority (CA). The role of a CA is to guarantee that a user is who he claims to be and is entitled to own his/her certificate. It is up to the user to discover which CA he/she should contact. In general CAs are organized geographically and by research institute. Each CA has its own procedure to release certificates.

The updated list of recognized CAs is available at the URL:

```
http://lcg-registrar.cern.ch/pki_certificates.html
```

To be used in the LCG-1 Grid, the certificate must be in PEM format. An important property of a certificate is the *subject*, a string containing information about the user. A typical example is:

```
/O=Grid/O=CERN/OU=cern.ch/CN=John Doe
```

If the certificate is in PKCS12 format (extension .p12), it can be converted to PEM (extension .pem) in this way:

```
$ openssl pkcs12 -nocerts \
  -in my_cert.p12 \
  -out userkey.pem
$ openssl pkcs12 -clcerts -nokeys
  -in cert.p12
  -out usercert.pem
```

where:

userkey.pem is the path to the private key file (**This should be set with permissions so that only the owner can read it.**) (e.g. `chmod 400 userkey.pem`)

`usercert.pem` is the path to your certificate file. (e.g. `chmod 444 usercert.pem`)

`my_cert.p12` is the path for the output PKCS12 format file to be created.

It is very useful to upload the user certificate into a WEB browser or e-mailer to be able to sign requests digitally. Browsers (including Internet Explorer and Netscape) use a certificate format different than the one used by the LCG-1 grid software. Browsers require a format called PKCS12 whereas grid software uses PEM format. If the certificate was issued to a user in PEM format then on a machine with the *openssl* package installed the following command can be used to convert from PEM to PKCS12 :

```
openssl pkcs12 -export -inkey userkey.pem -in usercert.pem \  
              -out my_cert.p12 -name "My certificate"
```

Where:

`userkey.pem` is the path to the private key file (**This should be set with permissions so that only the owner can read it.**) (e.g. `chmod 400 userkey.pem`)

`usercert.pem` is the path to your certificate file. (e.g. `chmod 444 usercert.pem`)

`my_cert.p12` is the path for the output PKCS12 format file to be created.

`"My certificate"` is an optional name which can be used to select this certificate in the browser after the user has uploaded it if the user has more than one.

The user certificate must be imported to a web browser to perform the next step. Instructions about how to load a certificate into a WWW browser are available at

http://lcg-registrar.cern.ch/load_certificates.html

4.2. REGISTERING WITH LCG

Before a user can use the LCG-1 service, registration of some personal data with the LCG registration server (hosted at CERN) plus some additional steps are required. For detailed information please visit the following URL:

<http://lcg-registrar.cern.ch/>

to actually register oneself to the LCG-1 service, it is necessary to use a WWW browser with the user certificate for the request to be properly authenticated.

4.3. VIRTUAL ORGANIZATIONS

A second requirement for the user is to belong to a *Virtual Organization* (VO). A VO is an entity, which corresponds typically to a particular organization or group of people in the real world, whose membership grants specific privileges to the user. For example, belonging to the ATLAS VO will allow the user to read the ATLAS files or to exploit resources reserved to the ATLAS collaboration.

Entering the VO of an experiment usually requires being a member of the collaboration; the user must comply with the rules of the VO relevant to him to gain membership. Of course, it is also possible to be expelled from a VO when the user fails to comply with these rules.

It is not possible to access the LCG-1 Grid without being member of any VO. However, it is possible to belong to more than one VO at the same time. In that case, the user must choose, when submitting a job, what is VO context for the specific job: it cannot exploit the advantage of being in two VOs at the same time.

A complete list of the VOs accepted by LCG-1 is available at the URL:

```
http://lcg-registrar.cern.ch/virtual_organization.html
```

4.4. SETTING UP THE USER ACCOUNT

To access the LCG-1 Grid, the user must also have an account on a LCG-1 User Interface (UI).

To obtain such an account, a local system administrator must be contacted. The official list of LCG sites is available at the URL:

```
http://cern.ch/grid-deployment/cgi-bin/index.cgi?var=lcg1Status
```

As an alternative, the user can install the UI software on his/her machine (see the Installation and Administration Guide).

Once the account has been created, the user certificate must be installed. For that, it is necessary to create a directory named `.globus` under the user home directory and put there the user certificate and key files naming them `usercert.pem` and `userkey.pem` respectively, with permissions 0400 for the latter, and 0444 for the former.

4.5. CHECKING A CERTIFICATE

To verify that a certificate is not corrupted and print some information about it, the Globus command `grid-cert-info` can be used from the user's UI account. The `openssl` command can be used instead to verify the validity of a certificate with respect to the certificate of the certification authority that issued it.

Example 4.5.1. (printing information on a user certificate)

With the certificate properly installed in the `$HOME/.globus` directory of the user's UI account, issue the command:

```
$ grid-cert-info
```

If the certificate is properly formed, the output will be something like:

```
Certificate:
```

Data:

Version: 3 (0x2)
Serial Number: 5 (0x5)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=CH, O=CERN, OU=cern.ch, CN=CERN CA

Validity

Not Before: Sep 11 11:37:57 2002 GMT
Not After : Nov 30 12:00:00 2003 GMT

Subject: O=Grid, O=CERN, OU=cern.ch, CN=John Doe

Subject Public Key Info:

Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:ab:8d:77:0f:56:d1:00:09:b1:c7:95:3e:ee:5d:
c0:af:8d:db:68:ed:5a:c0:17:ea:ef:b8:2f:e7:60:
2d:a3:55:e4:87:38:95:b3:4b:36:99:77:06:5d:b5:
4e:8a:ff:cd:da:e7:34:cd:7a:dd:2a:f2:39:5f:4a:
0a:7f:f4:44:b6:a3:ef:2c:09:ed:bd:65:56:70:e2:
a7:0b:c2:88:a3:6d:ba:b3:ce:42:3e:a2:2d:25:08:
92:b9:5b:b2:df:55:f4:c3:f5:10:af:62:7d:82:f4:
0c:63:0b:d6:bb:16:42:9b:46:9d:e2:fa:56:c4:f9:
56:c8:0b:2d:98:f6:c8:0c:db

Exponent: 65537 (0x10001)

X509v3 extensions:

Netscape Base Url:

<http://home.cern.ch/globus/ca>

Netscape Cert Type:

SSL Client, S/MIME, Object Signing

Netscape Comment:

For DataGrid use only

Netscape Revocation Url:

<http://home.cern.ch/globus/ca/bc870044.r0>

Netscape CA Policy Url:

<http://home.cern.ch/globus/ca/CPS.pdf>

Signature Algorithm: md5WithRSAEncryption

30:a9:d7:82:ad:65:15:bc:36:52:12:66:33:95:b8:77:6f:a6:
52:87:51:03:15:6a:2b:78:7e:f2:13:a8:66:b4:7f:ea:f6:31:
aa:2e:6f:90:31:9a:e0:02:ab:a8:93:0e:0a:9d:db:3a:89:ff:
d3:e6:be:41:2e:c8:bf:73:a3:ee:48:35:90:1f:be:9a:3a:b5:
45:9d:58:f2:45:52:ed:69:59:84:66:0a:8f:22:26:79:c4:ad:
ad:72:69:7f:57:dd:dd:de:84:ff:8b:75:25:ba:82:f1:6c:62:
d9:d8:49:33:7b:a9:fb:9c:1e:67:d9:3c:51:53:fb:83:9b:21:

c6:c5

The `grid-cert-info` command takes many options. Use the `-help` for a full list. For example, the `-subject` option returns the certificate subject:

```
$ grid-cert-info -subject
/O=Grid/O=CERN/OU=cern.ch/CN=John Doe
```

Example 4.5.2. (Verify a user certificate)

Just issue from the UI the command:

```
$ openssl verify -CApath /etc/grid-security/certificates \
 ~/.globus/usercert.pem
```

and if the certificate is valid, the output will be:

```
/home/doe/.globus/usercert.pem: OK
```

IF the certificate of the CA which issued the user certificate is not found in `-CApath`, an error message like this will appear:

```
usercert.pem: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
error 20 at 0 depth lookup:unable to get local issuer certificate
```

4.6. PROXY CERTIFICATES

At this point, the user is able to generate a *proxy certificate*. A proxy certificate is a delegated user credential that authenticates the user in every secure interaction, and has a limited lifetime: in fact, it prevents having to use one's own certificate, which could compromise its safety.

The command to create a proxy certificate is `grid-proxy-init`, which prompts for the user pass phrase, as in this example:

Example 4.6.1. (Create a proxy certificate)

To create a proxy certificate, issue the command

```
$ grid-proxy-init
```

If the command is successful, the output will be like

```
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until: Tue Jun 24 23:48:44 2003
```

and the proxy certificate will be written in `/tmp/x509up_u<uid>`, where `<uid>` is the Unix UID of the user.

If the user gives a wrong pass phrase, the output will be

```
ERROR: Couldn't read user key. This is likely caused by
either giving the wrong passphrase or bad file permissions
key file location: /home/doe/.globus/userkey.pem
```

Use `-debug` for further information.

If the proxy certificate file cannot be created, the output will be

```
ERROR: The proxy credential could not be written to the output file.
```

```
Use -debug for further information.
```

If the user certificate files are missing, or the permissions of `userkey.pem` are not correct, the output is

```
ERROR: Couldn't find valid credentials to generate a proxy.
```

```
Use -debug for further information.
```

In case of success, the proxy certificate is created in `/tmp`, unless the environment variable `X509_USER_PROXY` is defined (e.g. `X509_USER_PROXY=$HOME/.globus/proxy`), in which case the proxy with that file name will be created, if possible.

By default, the proxy has a lifetime of 12 hours; to specify a different lifetime, the `-valid H:M` option can be used (Proxy is valid for H hours and M minutes - default:12:00; the old option `-hours` is deprecated). When a proxy certificate has expired, it becomes useless and a new one has to be created with `grid-proxy-init`. Longer lifetimes imply bigger security risks, though. Use the option `-help` for a full listing of options.

It is also possible to print information about an existing proxy certificate, or to destroy it before its expiration, as in the following examples.

Example 4.6.2. (Printing information on a proxy certificate)

To print information about a proxy certificate, for example, the subject or the time left before expiration, give the command:

```
$ grid-proxy-info
```

The output, if a valid proxy exists, will be similar to

```
subject  : /O=Grid/O=CERN/OU=cern.ch/CN=John Doe/CN=proxy
issuer   : /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
type     : full
strength : 512 bits
path     : /tmp/x509up_u7026
timeleft : 11:59:56
```

If a proxy certificate does not exist, the output is:

```
ERROR: Couldn't find a valid proxy.
```

```
Use -debug for further information.
```

Example 4.6.3. (Destroying a proxy certificate)

To destroy an existing proxy certificate before its expiration, it is enough to do

```
$ grid-proxy-destroy
```

If no proxy certificate exists, the result will be

```
ERROR: Proxy file doesn't exist or has bad permissions
Use -debug for further information.
```

Known limitations: A person with administrator privileges on a machine can steal

proxies and run jobs on the Grid.

4.7. ADVANCED PROXY MANAGEMENT

The proxy certificates created as described in the previous section have a disadvantage: if the job does not finish before the proxy expires, it is aborted. This is clearly a problem if, for example, the user must submit a number of jobs that take a lot of time to finish: he should create a proxy certificate with a very long lifetime, with increased security risks.

To overcome this limit, there is the Proxy credential repository system, which allows the user to create and store on a dedicated server a long-term proxy certificate. The WMS will then be able to use this long-term proxy to periodically renew the proxy for a submitted job before it expires and until the job ends (or the long-term proxy expires).

To see if an LCG-1 site has a Proxy Server, please refer to:

```
http://cern.ch/grid-deployment/cgi-bin/index.cgi?var=lcg1Status
```

ATTENTION!!! It is worth noting that proxy renewal is triggered by the WMS at 3/4 of the actual proxy lifetime. Thus if the initial proxy is very short, the proxy renewal might be triggered a bit too late, after the job has failed with the following error (at the moment, GRIDMANAGER_MINIMUM_PROXY_TIME is 600 seconds):

```
Status Reason: Got a job held event, reason: Globus error 131: the user
proxy expired (job is still running)
```

It is therefore recommended to always generate initial proxies with a lifetime longer than 600 seconds (=10 minutes).

Example 4.7.1. (Creating a long-term proxy and store in a MyProxy server)

To create and store a long-term proxy certificate, the user must do, for example:

```
$myproxy-init -s <host_name> -d -n
```

The output is similar to:

```
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
```

```
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until: Thu Jul 17 18:57:04 2003
A proxy valid for 168 hours (7.0 days) for user
/O=Grid/O=CERN/OU=cern.ch/CN=John Doe now exists on
lxshare0207.cern.ch.
```

where `<host_name>` is the hostname of the machine where a Proxy Server runs. By default, the long-term proxy lasts for one week and the proxy certificates created from it last 12 hours. These lifetimes can be changed using the `-c` and the `-t` option, respectively.

If the `-s <host_name>` option is missing, the command will try to use the `MYPROXY_SERVER` environment variable to determine the Proxy server.

ATTENTION! If the hostname of the Proxy Server is wrong, or the service is unavailable, the output will be similar to

```
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until: Wed Sep 17 12:10:22 2003
Unable to connect to adc0014.cern.ch:7512
```

Where only the last line reveals that an error occurred.

Example 4.7.2. (Retrieving info about a long-term proxy)

To get information about a long-term proxy stored in a Proxy server, one must give the command:

```
$ myproxy-info -s <host_name> -d
```

The output is similar to

```
username: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
owner: /O=Grid/O=CERN/OU=cern.ch/CN=John Doe
timeleft: 167:59:48 (7.0 days)
```

ATTENTION! An important thing to remember is that the user must have a valid proxy certificate on the UI, created with `grid-proxy-init`, to successfully interact with his long-term certificate on the Proxy server.

Example 4.7.3. (Deleting a long-term proxy)

Deleting a stored long-term proxy is achieved by doing:

```
$ myproxy-destroy -s <host_name> -d
```

And the output is:

```
Default MyProxy credential for user /O=Grid/O=CERN/OU=cern.ch/CN=John
Doe was successfully removed.
```

Also in this case, a valid proxy certificate must exist for the user on the UI.

5. JOB MANAGEMENT

5.1. THE COMMAND LINE INTERFACE

In this section, all the commands used to submit and cancel jobs, query their status and retrieving their output, are described. The language used to describe a job, called *Job Description Language (JDL)* is also explained.

5.1.1. Job submission

To submit a job to the LCG-1 Grid, the user must have a valid proxy certificate and use the following command:

```
$ edg-job-submit <jdl_file>
```

where <jdl_file> is a file containing the job description, usually with extension .jdl.

Example 5.1.1.1. (Submitting a simple job)

Create a file test.jdl with this content:

```
Executable = "/bin/hostname";
StdOutput = "std.out";
StdError = "std.err";
OutputSandbox = {"std.out","std.err"};
```

It describes a simple job consisting of executing /bin/hostname. Standard output and error are directed to the files std.out and std.err respectively, which are then transferred back to the user interface after the job is finished, as they are in the output sandbox. The job is submitted by issuing:

```
$ edg-job-submit test.jdl
```

If the submission is successful, the output is similar to:

```
*****
                                JOB SUBMIT OUTCOME
The job has been successfully submitted to the Network Server.
Use edg-job-status command to check job current status. Your job
identifier (edg_jobId) is:

- https://lxshare0234.cern.ch:9000/rIBubkFFKhnSQ6CjiLUY8Q
*****
```

The command returns to the user the job identifier (jobId), which defines univocally the job and can be used to perform further operations on the job, like interrogating the system about its status, or cancelling it. The format of the jobId is:

```
https://Lbserver_address[:port]/unique_string
```

where `unique_string` is guaranteed to be unique and `Lbserver_address` is the address of the logging and bookkeeping server for the job, and usually (but not necessarily) is also the Resource Broker. The `jobId` does NOT identify a web page.

If the command returns the following error:

```
**** Error: API_NATIVE_ERROR ****
Error while calling the "NSClient::multi" native api
AuthenticationException: Failed to establish security context...

**** Error: UI_NO_NS_CONTACT ****
Unable to contact any Network Server
```

It means that there are authentication problems between the UI and the network server (check your proxy or have the site admin check the certificate of the server).

Many options are available to `edg-job-submit`.

The useful `-o <file_path>` option allows users to specify a file to which the `jobId` of the submitted job will be appended. This file can be given to other job management commands to perform operations on more than one job with a single command.

The user can specify his virtual organization with the `-vo <vo_name>` option; otherwise the default VO specified in the file `$EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf` is used.

The `-r <CE_Id>` option is used to directly send a job to a particular CE; the drawback is that the **Brokerinfo** functionality (see later) will not be available. Similarly, the `-i <file_path>` allows to specify a list of CEs from where the user will have to choose a target CE interactively. A CE is identified by a string like

```
host.domain:2119/jobmanager-<queue>
```

as, for example, `adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite`.

Example 5.1.1.2. (Listing computing elements that match the job description)

It is possible to see which CEs are eligible to run a job specified by a given JDL file using the command `edg-job-list-match`:

```
$ edg-job-match test.jdl
Connecting to host lxshare0380.cern.ch, port 7772

*****
COMPUTING ELEMENT IDs LIST
The following CE(s) matching your job requirements have been found:

*CEId*
adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite
```

```
adc0015.cern.ch:2119/jobmanager-lcgpbs-long
adc0015.cern.ch:2119/jobmanager-lcgpbs-short
*****
```

The `-o <file path>` option can be used to store the CE list on a file, which can later be used with the `-i <file path>` option of `edg-job-submit`.

5.1.2. Job description language (JDL)

The job description language is used to write job description files, which contain a description of the job characteristics and constraints. The JDL syntax is based on the Condor ClassAds library, and it consists of statements like:

```
attribute = value;
```

ATTENTION!!! The JDL is sensitive to blank characters and tabs. No blank characters or tabs should follow the semicolon at the end of a line.

In a job description file, some attributes are mandatory, while some others are optional. Essentially, one must at least specify the name of the executable, the files where to write the standard output and the standard error of the job (they can even be the same file). For example:

```
Executable = "test.sh";
StdOutput = "std.out";
StdError = "std.err";
```

If needed, arguments to the executable can be passed:

```
Arguments = "hello 10";
```

if the argument list contains quoted strings, the quotes must be escaped with a backslash (e.g. `Arguments = "\"hello\" 10"`). In general, special characters such as `&`, `|`, `>`, `<` are only allowed if specified inside a quoted string or preceded by triple `\`. The character ``` cannot be specified in the JDL.

The standard input can be similarly specified:

```
StdInput = "std.in";
```

Then, the files to be transferred between the UI and the WN before ("input sandbox") and after ("output sandbox") the job execution can be specified:

```
InputSandbox = {"test.sh"};
OutputSandBox = {"std.out","std.err"};
```

Wildcards are allowed only in the `InputSandbox` attribute. The list of files in the `InputSandbox` is specified relatively to the current working directory. Absolute paths cannot be specified in the `OutputSandbox`.

The environment of the job can be modified using the `Environment` attribute. For example:

```
Environment = {"CMS_PATH=$HOME/cms",
               "CMS_DB=$CMS_PATH/cmdb"}
```

If the job requires some files stored in an LCG storage element, the `InputData` attribute can be used to make the resource broker select a CE as close as possible to the files. The `OutputData` attribute, similarly, can be used to automatically have any output data files copied to a SE, while the `OutputSE` attribute forces a job to be run on a CE "close" to a given SE.

Example 5.1.2.1. (Specifying input data in a job)

If the user job needs to read two files (identified by a logical file name or by their GUID) and wants to run "close" to them, the job description file must contain a line like:

```
InputData = {"lfn:doe/prod/kin_1",  
            "guid:136b48a64-4a3d-87ud-3bk5-8gnn46m49f3"};
```

In addition, the user must declare which protocols his application is able to use to read the files:

```
DataAccessProtocol = {  
    "file", "gridftp"};
```

The only supported protocols are *file* (meaning that the files must be in an NFS-mounted directory), *gridftp* (the GSI version of ftp) and *rfile*.

The job will then be sent to the CE with the best "rank" (which is a user-definable measurement of the CE "goodness"), between all the CEs satisfying all the job requirements and having the maximum number of file replicas on a SE "close" to them.

Example 5.1.2.2. (Specifying a storage element)

The user can force the job to run close a specific storage element using the attribute `OutputSE`. For example:

```
OutputSE = "lxshare0291.cern.ch";
```

The Resource Broker will not abort the job if there is no CE close to the `OutputSE` specified by the user. The RB will try to find resources close to such SE but if the CE cannot be found the job will run somewhere else.

To express any kind of requirement on the resources where the job can run, there is the `Requirements` attribute. Its value is a Boolean expression that must evaluate to true for a job to run on that specific CE.

Example 5.1.2.3. (Specifying a requirement on the CE)

Let us suppose that the user wants to run on a CE using PBS and whose WNs have at least two CPUs. He will write then in the job description file:

```
Requirements = other.GlueCEInfoLRMSType == "PBS" &&  
              Other.GlueCEInfoTotalCPUs > 1;
```

For example, one can ask the WMS to send a job to a particular CE with an expression like:

```
Requirements = other.GlueCEUniqueID ==  
              "lxshare0286.cern.ch:2119/jobmanager-pbs-short";
```

As a general rule, requirements on attributes of a CE are written prefixing "other." to the attribute name in the Information System schema.

Example 5.1.2.4. (Specifying a requirement using wildcards)

It is also possible to use wild cards (actually regular expressions) in expressing a requirement. Let us suppose for example that the user wants all this jobs to run on CEs in the domain cern.ch. This can be achieved putting in the JDL file the following expression:

```
Requirements = RegExp("*.cern.ch", other.GlueCEUniqueId);
```

The opposite can be required just by using

```
Requirements = (!RegExp("*.cern.ch", other.GlueCEUniqueId));
```

A detailed description of the JDL syntax is out of the scope of this guide, and can be found in [R7].

In general, all CE attributes present in the Information System can be used in the Requirements or Rank value part in the JDL file. For a list of GLUE attributes, check later in the Section dedicated to the Information System.

In order to specify requirements on entities different from a CE but in a relation to a CE, a mechanism called "gangmatching" is supported. Rather than explaining it in detail, a typical example is described.

Example 5.1.2.5. (Specifying a requirement on the close SE)

To ensure that the job runs on a CE with, for example, at least 200 MB of free disk space in a close SE, the following JDL expression can be used:

```
Requirements = anyMatch(other.storage.CloseSEs,  
    target.GlueSAStateAvailableSpace > 20971520);
```

Another way to specify the VO of the user is using the `VirtualOrganisation`¹ attribute, as for example:

```
VirtualOrganisation = "cms";
```

Whose value is anyway superseded by the `-vo` option of `edg-job-submit`.

A JDL attribute called `RetryCount` can be used to specify how many times the WMS must try to resubmit a job if it fails due to some LCG component (that is, not the job itself). The default value (if any) is defined in the file `$EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf`.

The `MyProxyServer` attribute indicates the MyProxy server containing the user's long-term proxy that the WMS must use to renew the proxy certificate when it is about to expire.

The choice of the CE where to execute the job, among all the ones satisfying the requirements, is based on the "rank" of the CE, that is, a quantity expressed as a floating-point number; the CE with the highest rank is the one selected.

¹ A common error is to write `VirtualOrganization`: it will not work!

The user can define the rank with the `Rank` attribute as a function of the CE attributes, like in the following (which is also the default definition):

```
Rank = other.GlueCEStateFreeCPUs;
```

5.1.3. Checkpointable jobs

Checkpointable jobs are not supported at present.

5.1.4. Interactive jobs

Interactive jobs are not supported at present.

5.1.5. Job operations

After a job is submitted, it is possible to see its status and its history, to cancel it, and to retrieve its output once it is finished. The following examples explain how.

Example 5.1.5.1 (see the status of a job)

Given a submitted job whose job identifier is `<jobId>`, the command is:

```
$ edg-job-status <jobId>
```

an example of a possible output is

```
*****
BOOKKEEPING INFORMATION:

Printing status info for the Job :
https://lxshare0234.cern.ch:9000/X-ehTxfdlXxSoIdVLS0L0w
Current Status:      Ready
Status Reason:      unavailable
Destination:        lxshare0277.cern.ch:2119/jobmanager-pbs-infinite
reached on:         Fri Aug  1 12:21:35 2003
*****
```

The possible states in which a job can be found are described in section **Error! Reference source not found.**

Many job identifiers can be given as arguments, i.e.:

```
edg-job-status <jobId1> ... <jobIdN>
```

The option `-i <file path>` can be used to specify a file with a list of job identifiers (like the one produced by the `-o` option of `edg-job-submit`), in which case the command asks interactively to the user the status of which job(s) must print:

```
$ edg-job-status jobs.list
```

```
-----
1 : https://lxshare0234.cern.ch:9000/UPBqN2s2ycxt1TnuU3kzEw
2 : https://lxshare0234.cern.ch:9000/8S6IwPW33AhyxhkSv8Nt9A
3 : https://lxshare0234.cern.ch:9000/E9R0Y14J7qgsq7FYTnhmsA
4 : https://lxshare0234.cern.ch:9000/Tt80pBn17AFPJyUSN9Qb7Q
a : all
q : quit
```

Choose one or more `edg_jobId(s)` in the list - [1-4]all:

Subsets of jobs can be selected (e.g. 1-2,4).

With the option `-o <file path>` the command output can be written to a file.

NOTE: the `--all` option is not supported in LCG-1.

Example 5.1.5.2 (Cancelling a job)

A job can be cancelled before it ends using the command `edg-job-cancel`.

This command requires as arguments one or more job identifiers. For example:

```
$ edg-job-cancel \  
https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog \  
https://lxshare0234.cern.ch:9000/C6n5Hq1ex9-wF2t05qe8mA  
  
Are you sure you want to remove specified job(s)? [y/n]n :y  
===== edg-job-cancel Success=====  
  
The cancellation request has been successfully submitted for the  
following job(s)  
- https://lxshare0234.cern.ch:9000/dAE162is6EStca0VqhVkog  
- https://lxshare0234.cern.ch:9000/C6n5Hq1ex9-wF2t05qe8mA
```

All the command options work exactly as in `edg-job-status`.

NOTE: the `--all` option is not supported in LCG-1.

Example 5.1.5.3 (retrieve the output of a job)

After the job has finished (is is the "Done" status), its output can be copied to the UI with the command `edg-job-get-output`, which takes a list of job as argument. For example:

```
$ edg-job-get-output \  
https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg  
  
Retrieving files from host lxshare0234.cern.ch  
  
*****  
JOB GET OUTPUT OUTCOME  
  
Output sandbox files for the job:  
- https://lxshare0234.cern.ch:9000/snPegp1YMJcnS22yF5pFlg  
have been successfully retrieved and stored in the directory:  
/tmp/jobOutput/snPegp1YMJcnS22yF5pFlg  
  
*****
```

By default, the output is stored under `/tmp`, but it is possible to specify in which directory to save the output using the `-d <path_name>` option.

All command options work exactly as in `edg-job-status`.

NOTE: the `--all` option is not supported in LCG-1.

5.1.6. Advanced command options

All the `edg-job-*` commands read some configuration files which the user can edit, if he is not satisfied with the default ones.

The main configuration file is located by default at `$EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf`, and sets, among other things, the default VO, the default location for job outputs and command log files and the default values of mandatory JDL attributes. Another configuration file can be used by setting the value of the environment variable `$EDG_WL_UI_CONFIG_VAR` to the file path, or by specifying the file in the `--config <file>` option of the `edg-job-*` commands (which takes precedence).

In addition, VO-specific configurations are defined by default in the file `$EDG_WL_LOCATION/etc/<vo>/edg_wl_ui.conf`, consisting essentially in the list of Network servers, MyProxy servers and LB servers accessible to that VO. A different file can be specified using the variable `$EDG_WL_UI_CONFIG_VO` or the `--config-vo <file>` option of the `edg-job-*` commands.

Example 5.1.5.4 (change the default VO)

A user can change the default VO for him by performing the following steps:

- a) Make a copy of the file `$EDG_WL_LOCATION/etc/edg_wl_ui_cmd_var.conf`, for example to `~/my_ui.conf`;
- b) Edit `~/my_ui.conf` and change this line:
`DefaultVo = "cms";`
 (if, for example, he wants to set the CMS VO as default);
- c) Define in the shell configuration script (`$HOME/.bashrc` for bash and `$HOME/.cshrc` for csh/tcsh) the environment variable


```
setenv EDG_WL_UI_CONFIG_VAR ~/my_ui.conf      (bash)
export EDG_WL_UI_CONFIG_VAR=~/my_ui.conf    ((t)csh)
```

The `--log <file>` option allows to define the log file; the default log file is named `<command_name>_<UID>_<PID>_<date_time>.log` and it is found in the directory specified in the configuration file.

The `--noint` option skips all interactive questions and prints all warning and error messages to a log file.

The `--help` and `--version` options are self-explanatory.

5.1.7. The BrokerInfo

The BrokerInfo file is a mechanism by which the user job can access, at execution time, certain information concerning the job, for example the name of

the CE, the files specified in the InputData attribute, the Storage Elements where they can be found, etc.

The BrokerInfo file is created in the job working directory (that is, the current directory on the WN for the executable) and is named .BrokerInfo. Its syntax is based on Condor ClassAds and the information contained is not easy to read; however, it is possible to get it by means of a command line interface, described in the following.

The `edg-brokerinfo` command has the syntax:

```
edg-brokerinfo [-v] [-f filename] function [parameter] [parameter] ...
```

where `function` is one of the following:

- `getCE`: returns the name of the CE the job is running on;
- `getDataAccessProtocol`: returns the protocol list specified in the `DataAccessProtocol` JDL attribute;
- `getInputData`: returns the file list specified in the `InputData` JDL attribute;
- `getSEs`: returns the list of the storage elements with contain a copy of at least one file among those specified in `InputData`;
- `getCloseSEs`: returns a list of the storage elements close to the CE;
- `getSEMOUNTPOINT <SE>`: returns the NFS mount point on the WN of the storage element `<SE>` if it is NFS-mounted;
- `getSEFreeSpace <SE>`: returns the free space on `<SE>`;
- `getLFN2SFN <LFN>`: returns the storage file name of the file specified by `<LFN>`, where `<LFN>` is a logical file name of a GUID specified in the `InputData` attribute;
- `getSEProtocols <SE>`: returns the list of the protocols available to transfer data in the storage element `<SE>`;
- `getSEPort <SE> <Protocol>`: returns the port number used by `<SE>` for the data transfer protocol `<Protocol>`;
- `getVirtualOrganization`: returns the name of the VO specified in the `VirtualOrganization` JDL attribute.
- `getAccessCost`: not supported at present.

The `-v` option produced a more verbose output, and the `-f filename` option tells the command to parse the BrokerInfo file specified by `<filename>`. In fact, by default, the command tries to parse the file `$EDG_WL_RB_BROKERINFO/.BrokerInfo`, where `$EDG_WL_RB_BROKERINFO` coincides with the working directory of the job.

6. DATA MANAGEMENT

In what follows we describe the EDG Data Management Tools, we list some use cases and example usage. Low level tools like `globus-url-copy` should only be used in case of problems and anyway by system administrators only and not by LCG-1 Grid users. We will describe them at the end of this Section but as a reference only. Also, we describe details on how to physically access a file available on a SE.

The Data Management Tools available for uploading files to the Grid, replicating data and localizing the best replica to use are:

| | |
|---|--------------|
| <code>edg-replica-manager</code> | client tools |
| <code>edg-replica-optimization</code> | client tools |
| <code>edg-local-replica-catalog</code> | client tools |
| <code>edg-replica-metadata-catalog</code> | client tools |

Details on how to use the client tools mentioned above can be found in [R9],[R10],[R11],[R12].

Lower level tools allow users to perform some actions on the SE. These are:

| | |
|---------------------------------|--|
| <code>edg-gridftp-exists</code> | to check the existence of a file or directory on SE. |
| <code>edg-gridftp-ls</code> | to list a directory on SE. |
| <code>edg-gridftp-mkdir</code> | to create a directory on SE. |
| <code>edg-gridftp-rename</code> | to rename a file on SE. |
| <code>edg-gridftp-rm</code> | to remove a file on SE. |
| <code>edg-gridftp-rmdir</code> | to remove a directory on SE. |

The commands `edg-gridftp-rename`, `edg-gridftp-rm`, and `edg-gridftp-rmdir` should be used with extreme care and only in case of serious problems. In fact these commands do not interact with any of the catalogues and therefore they can compromise the consistency/coherence of the information contained. To obtain help on this commands use the option `--usage` or `--help`.

6.1. THE EDG-REPLICA-MANAGER CLIENT TOOLS

The EDG replica manager client tools allow users to copy files from UI, CE and WN to a SE, to register entries in the RLS and replicate files between SEs.

In what follows we give some usage example.

For a description of the command line tool, enter the following command:

```
$ edg-rm --help
usage: edg-replica-manager [options] command [command-options]
  -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
  --timing <file>        write timing performance log information to the
                        specified file
  --vo <VO>              set Virtual Organization
  -v,--verbose            print additional information while executing
Commands: [...]
```

Example 6.1.1. (Uploading a file from the UI to the Grid)

In the following example we describe how to upload a file to the Grid, i.e. how to transfer it from the local machine to a Storage Element where it is then stored permanently. Before one can upload the file to a Storage Element, one should know:

- where is a potential SE that can be used
- once an SE is found, where the file can be written

There are several ways to find out about this SE specific information: one can either use the EDG Replica Manager command `printInfo` or query the information system directly. Here, we describe the simplest way of getting this information from the Replica Manager.

```
$ edg-rm --vo=cms printInfo
```

Returns all Ces and SEs that the Replica Manager retrieves from the Information System. A typical output is as follows:

```
VO used                : cms
default SE             : lxshare0236.cern.ch
default CE             : lxshare0235.cern.ch
Info Service class    : org.edg.data.reptor.info.InfoServiceMDS

RMC endpoint          : http://rlscert01.cern.ch:7777/edg-replica-metadata-catalog/
                        services/edg-replica-metadata-catalog
LRC endpoint          : http://rlscert01.cern.ch:7777/edg-replica-location/services
                        /edg-local-replica-catalog
```

ROS endpoint: no information found: No Service found edg-replica-
optimization

List of CE ID's: lxshare0235.cern.ch:2119/jobmanager-pbs-infinite
lxshare0235.cern.ch:2119/jobmanager-pbs-long
lxshare0235.cern.ch:2119/jobmanager-pbs-medium
lxshare0235.cern.ch:2119/jobmanager-pbs-short

[...]

CE at infinite :

name : infinite
ID: lxshare0235.cern.ch:2119/jobmanager-pbs-infinite
closeSEs : lxshare0236.cern.ch
VOs : alice,atlas,cms,lhcb,wpsix,iteam,lcg

[...]

List of SE ID's : lxshare0236.cern.ch
lxshare0278.cern.ch
lxshare0291.cern.ch
lxshare0287.cern.ch

SE at LCGCERTTB1 :

name : LCGCERTTB1
host : lxshare0236.cern.ch
type : disk
accesspoint : /flatfiles/LCG-CERT-SE01
VOs : alice,atlas,cms,iteam,lcg,lhcb,wpsix
VO directories : alice:/alice,atlas:/atlas,cms:/cms,iteam:/iteam,
lcg:/lcg,lhcb:/lhcb,wpsix:/wpsix
protocols : file,gsiftp,rfio

[...]

In order to find all SEs, their access point and the VO directories the user can issue the following command:

```
$ edg-rm --vo=cms printInfo | grep -e SE -e host -e accesspoint \  
-e 'VO directories' | grep -v closeSEs | grep -v "List of SE"
```

```
default SE : lxshare0236.cern.ch
```

SE at LCGCERTTB1 :

```
host : lxshare0236.cern.ch  
accesspoint : /flatfiles/LCG-CERT-SE01  
VO directories: alice:/alice,atlas:/atlas,cms:/cms,iteam:/iteam,  
lcg:/lcg,lhcb:/lhcb,wpsix:/wpsix
```

```
SE at LCGCERTTB2 :
    host : lxshare0278.cern.ch
    accesspoint : /flatfiles/LCG-CERT-SE02
    VO directories: alice:/alice,atlas:/atlas,cms:/cms,iteam:/iteam,
                  lcg:/lcg,lhcb:/lhcb,wpsix:/wpsix

SE at LCGCERTTB3 :
    host : lxshare0291.cern.ch
    accesspoint : /flatfiles/LCG-CERT-SE03
    VO directories: alice:/alice,atlas:/atlas,cms:/cms,iteam:/iteam,
                  lcg:/lcg,lhcb:/lhcb,wpsix:/wpsix

SE at LCGCERTTB4 :
    host : lxshare0287.cern.ch
    accesspoint : /flatfiles/LCG-CERT-SE04
    VO directories: alice:/alice,atlas:/atlas,cms:/cms,iteam:/iteam,
                  lcg:/lcg,lhcb:/lhcb,wpsix:/wpsix
```

The `printInfo` command does not return the free space on the SE. For this information we have to check the Information Service, as described later.

Let us suppose, for example, that we are interested in uploading a file to the SE `lxshare0287` and we are member of the VO `cms`. The file can be copied into the following directory on the specific SE: `/flatfiles/LCG-CERT-SE04/cms`. This directory can be obtained appending to the SE accesspoint the VO directory for the specific VO.

We can use the following command to copy the file to the SE and register it with the Replica Location Services:

```
$ edg-rm --vo=cms copyAndRegisterFile file:///home/flavia/my-file \
    -d srm://lxshare0287.cern.ch/flatfiles/LCG-CERT-SE04/cms/my-test-file
guid:56efa7d4-bf7e-11d7-b9c9-b31515e201f4
```

The Replica Manager then returns a GUID of the file back to the screen. In case one also wants to assign an LFN to the file, the option `-l lfn:my-lfn` can be used.

There are several ways to simplify the file transfer. For instance, one does not need to specify the file path explicitly but can leave it to the Replica Manager to determine the directory. It is thus sufficient to only specify the hostname (e.g. `lxshare0287`). Another possibility is to omit the destination completely, i.e. the option `-d` is not used, and then the Replica Manager finds a Storage Element where the file can be written to with minimal access latencies (using the Replica Optimization Service whenever available).

For further examples of the Replica Manager refer to the Replica Manager User Guide [R9].

Example 6.1.2. (Replicating a file)

Once a file is stored on an SE and registered with the Replica Location Service, the file can be replicated using the command

```
$ edg-rm --vo=cms replicateFile guid:56efa7d4-bf7e-11d7-b9c9-b31515e201f4\  
    -d lxshare0378.cern.ch  
No SE found in the Info Service with host lxshare0378.cern.ch  
$ edg-rm --vo=cms replicateFile guid:adb8e950-bf7e-11d7-a29c-fbbda1b7a6d1\  
    -d lxshare0291.cern.ch  
srm://lxshare0291.cern.ch/flatfiles/LCG-CERT-  
SE03/cms/generated/2003/07/26/filea94a9cf3-bf7f-11d7-9836-ee2aa3d34626
```

Example 6.1.3. (Listing replicas that are registered with a catalog)

Furthermore, the Replica Manager allows users to list all replicas that have been successfully registered with the Replica Location Service:

```
$ edg-rm --vo=cms listReplicas guid:56efa7d4-bf7e-11d7-b9c9-b31515e201f4  
srm://lxshare0287.cern.ch/flatfiles/LCG-CERT-SE04/cms/my-test-file
```

The tools `edg-local-replica-catalog` and `edg-replica-metadata-catalog` provide more functions for catalog interaction. We refer the reader to the next paragraph and to the respective User Guides for further information [R11, R12].

Example 6.1.4. (Deleting a replica)

Once a file is stored on a Storage Element and registered with a catalog, it can also be deleted again using the command

```
$ edg-rm --vo=cms deleteFile guid:adb8e950-bf7e-11d7-a29c-fbbda1b7a6d1 \  
    -s lxshare0287.cern.ch
```

Which removes the file from the file system and the catalog.

The command execute successfully even if the file was already deleted returning an exit status equal to zero.

Example 6.1.5. (Removing a SFN from the catalogues - LRC)

To unregister a SFN from the LRC catalogue without doing any operation on the relative SE, one can use the `unregisterFile` command:

```
$ edg-rm -i --vo=cms unregisterFile guid:0b4b344c-ad47-11d7-962b-  
8d5cc052f003 srm://lxshare0291.cern.ch/generated/2003/07/03/file151cc903-  
ad47-11d7-9e83-815888ab59a7
```

Example 6.1.6. (Listing all aliases/LFNs for a given file)

In order to list all aliases of a file, today the user has to invoke two tools: `edg-rm` and `edg-rmc` discussed later in this section.

```
$ edg-rm -i --vo=cms listGUID lfn:grodid-16998-0-rmstorm.txt
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
```

```
$ edg-rmc -i aliasesForGuid -h rlscert01.cern.ch -p 7777 --vo=cms
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
lfn:flavia-yet-another-alias, lfn:grodid-16998-0-rmstorm.txt
```

For more detailed examples on Replica Manager usage, Refer to the EDG Tutorial Handout [R13]. Note that there several more methods that are provided by the Replica Manager but we here we only provided a basic overview and refer to the Replica Manager User Guide [R9] for all other commands and details.

Example 6.1.7. (Listing an SE directory)

Through the replica manager client tools it is possible to list the content of a directory on an SE using either the `srn` or `gsiftp` protocols. However, as of today this command does not work properly and returns a garbage string.

```
$ edg-rm -i --vo=cms list gsiftp://lxshare0278.cern.ch/generated/2003/07
$ edg-rm -i --vo=cms list srm://lxshare0278.cern.ch/generated/2003/07
```

In order to overcome this problem the `edg-gridftp-ls` tool can be used instead. However, for `edg-gridftp-ls` to return the right info, the correct mountpoint of the directory on the SE for the given VO must be known. In order to achieve so, either the `edg-rm printInfo` command can be used or the Information System must be interrogated, as shown in the next sections of this User Guide.

```
$ edg-gridftp-ls --verbose gsiftp://lxshare0278.cern.ch/flatfiles/LCG-CERT-
SE02/cms/generated/2003/07/26
total 8
-rw-rw-r--      1 cms002      cms          4841 Jul 26 17:42 filea94a9cf3-bf7f-
11d7-9836-ee2aa3d34626
```

Example 6.1.8. (Accessing a Grid file from a job)

```
# Get the first input file name
infile=`edg-brokerinfo getInputData | cut -d " " -f 1`

# Get the first close SE
CloseSE=`edg-brokerinfo getCloseSEs | cut -d " " -f 1`
```

```
# Make sure the file is on the close SE and return a file: TURL
TURL=`edg-rm --vo=iteam gbf $infile -d $CloseSE -t file`
```

```
# Chop off the file: scheme
localfile=`echo $TURL | cut -d ":" -f 2`
```

6.2. THE EDG-LOCAL-REPLICA-CATALOG AND EDG-REPLICA-METADATA-CATALOG CLIENT TOOLS

The `edg-local-replica-catalog` and `edg-replica-metadata-catalog` client tools are low level tools that allow users to browse and directly manipulating the LRC and the RMC catalogues.

ATTENTION!!! It is very important to know that the client tools do not enforce any syntax checking on catalogue entries and therefore it is very easy to corrupt the content of the catalogue making it unusable by the `edg-replica-manager` client tools.

ATTENTION!!! Using these tools, a user can change the content of the catalogues making them inconsistent. For instance, a GUID can be removed from the RMC but not from the LRC making a file not addressable by its alias. Therefore a lot of care must be taken when using these tools.

When an entry is registered in the LRC and RMC catalogues by the `edg-replica-manager`, it can have a form as in the following examples:

For a GUID:

```
guid:38ed3f60-c402-11d7-a6b0-f53ee5a37e1d
```

For a SFN:

```
srm://lxshare0291.cern.ch/flatfiles/LCG-CERT-
SE03/cms/generated/2003/08/01/file3596e86f-c402-11d7-a6b0-
f53ee5a37e1d
```

For an LFN or User Alias:

```
lfn:sciaba/Tests/Prod_2/Test_cmkin_1240.dat
```

As you can see, entries must be prefixed by a lowercase `guid` or `srm` or `lfn`. In particular, the SFN must be a URL starting with `srm` followed by `://` and the full hostname of the SE holding the file.

Failure to comply with these rules results in corrupted catalogues and malfunctioning replica management.

The `edg-local-replica-catalog` and `edg-replica-metadata-catalog` client tools allow users to even perform wildcard operation on the catalogues.

All commands have the following syntax:

```
$ edg-rmc --help
usage: edg-replica-metadata-catalog [options] command [command-options]
  -h,--help          print help (if command is given, details on command)
  -i,--insecure      Connect in an insecure manner, i.e. not https.
  -v,--verbose       print additional information while executing
```

where `command` is specific to the client invoked.

We report here few usage examples.

Example 6.2.1. (Checking if an Alias/LFN exists)

```
$ edg-rmc -i aliasExists -h rlscert01.cern.ch -p 7777 --vo=cms \
          lfn:grodid-16998-0-rmstorm.txt
Alias exists : 'lfn:grodid-16998-0-rmstorm.txt'

$ edg-rmc -i aliasExists -h rlscert01.cern.ch -p 7777 --vo=cms \
          lfn:sciaba/Tests/Prod_2/Test_cmsim_1117.dat
Alias exists : 'lfn:sciaba/Tests/Prod_2/Test_cmsim_1117.dat'
$ edg-rmc -i aliasExists -h rlscert01.cern.ch -p 7777 --vo=cms \
          lfn:flavia/Tests/Prod_25/Test_cmsim_1110.dat
Alias does not exist : 'lfn:flavia/Tests/Prod_25/Test_cmsim_1110.dat'
```

Example 6.2.2. (Checking if a GUID exists in the RMC)

```
$ edg-rmc -i guidForAlias -h rlscert01.cern.ch -p 7777 --vo=cms \
          lfn:grodid-16998-0-rmstorm.txt
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003

edg-rmc -i guidExists -h rlscert01.cern.ch -p 7777 --vo=cms \
          guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
GUID exists : 'guid:0b4b344c-ad47-11d7-962b-8d5cc052f003'
```

Example 6.2.3. (Listing GUID to LFNs mapping for matching GUID wildcard in RMC)

```
$ edg-rmc -i mappingsByGuid -h rlscert01.cern.ch -p 7777 --vo=cms \  
    'guid:*344*'  
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003, lfn:grodid-16998-0-rmstorm.txt  
guid:344fea8b-be81-11d7-be56-e177b803a569,  
lfn:sciaba/Tests/Prod_1/Test_cmkin_5057.dat  
guid:59344db2-c41b-11d7-a2d6-d01c0ce321a1,  
lfn:sciaba/Tests/Prod_2/Test_cmsim_1242.dat
```

```
$ edg-rmc -i mappingsByGuid -h rlscert01.cern.ch -p 7777 --vo=cms '*'  
guid:01a96a9c-c41b-11d7-8632-9a6787b5c73a,  
lfn:sciaba/Tests/Prod_2/Test_cmsim_1218.dat  
guid:0310b5a0-b939-11d7-8244-99f9b21d8247,  
lfn:sciaba/Tests/Prod_1/Test_cmkin_5026.dat  
guid:03e78de3-b939-11d7-b397-dedb2e65d77a,  
lfn:sciaba/Tests/Prod_1/Test_cmkin_5027.dat  
guid:0493ea71-c41c-11d7-ad2c-f2b18bf8e5eb,  
lfn:sciaba/Tests/Prod_2/Test_cmsim_1285.dat  
guid:049739c1-c40c-11d7-90c9-cb0b7d008af7,  
lfn:sciaba/Tests/Prod_2/Test_cmkin_1292.dat  
guid:050479ff-c40c-11d7-8a67-b9a041ef05ac,  
lfn:sciaba/Tests/Prod_2/Test_cmkin_1293.dat  
[...]
```

Example 6.2.4. (Removing a given entry from RMC given the LFN)

```
$ GUID=`edg-rmc -i guidForAlias -h rlscms.cern.ch -p 7777 --vo cms  
lfn:cms/mu03_MB/cmkin/mu03_MB_77400311.ntpl`  
  
$ edg-rmc -i removeAlias -h rlscms.cern.ch -p 7777 --vo cms \  
$GUID lfn:cms/mu03_MB/cmkin/mu03_MB_77400311.ntpl
```

Example 6.2.5. (Adding a new Alias/LFN to a given GUID in RMC)

```
$ edg-rmc -i addMapping -h rlscms.cern.ch -p 7777 --vo=cms guid:00329.ntpl-  
1059676658-724-7425 lfn:cms/mu03_MB/cmkin/mu03_MB_77400329.ntpl
```

Equivalent examples can be given for edg-local-replica-catalog.

Example 6.2.6. (Checking if an SFN exists)

```
$ edg-lrc -i pfnExists -h rlscert01.cern.ch -p 7777 --vo=cms  
srm://lxshare0278.cern.ch/generated/2003/07/03/file090aldeb-ad47-11d7-962b-  
8d5cc052f003  
Pfn exists : 'srm://lxshare0278.cern.ch/generated/2003/07/03/file090aldeb-  
ad47-11d7-962b-8d5cc052f003'
```



Example 6.2.7. (Checking if a GUID exists in the LRC)

```
$ edg-rcm -i guidForAlias -h rlscert01.cern.ch -p 7777 --vo=cms lfn:grodid-16998-0-rmstorm.txt
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
$ edg-lrc -i guidExists -h rlscert01.cern.ch -p 7777 --vo=cms
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
GUID exists : 'guid:0b4b344c-ad47-11d7-962b-8d5cc052f003'
$ edg-lrc -i pfnsForGuid -h rlscert01.cern.ch -p 7777 --vo=cms
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003
srm://lxshare0236.cern.ch/generated/2003/07/03/file1f8dd1d0-ad47-11d7-b85d-f396da8af923, srm://lxshare0278.cern.ch/generated/2003/07/03/file090aldebad47-11d7-962b-8d5cc052f003,
srm://lxshare0291.cern.ch/generated/2003/07/03/file151cc903-ad47-11d7-9e83-815888ab59a7
```

Example 6.2.8. (Listing GUID to LFNs mapping for matching GUID wildcard in LRC)

```
$ edg-lrc -i mappingsByGuid -h rlscert01.cern.ch -p 7777 --vo=cms 'guid:*344c*'
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003,
srm://lxshare0236.cern.ch/generated/2003/07/03/file1f8dd1d0-ad47-11d7-b85d-f396da8af923
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003,
srm://lxshare0278.cern.ch/generated/2003/07/03/file090aldebad47-11d7-962b-8d5cc052f003
guid:0b4b344c-ad47-11d7-962b-8d5cc052f003,
srm://lxshare0291.cern.ch/generated/2003/07/03/file151cc903-ad47-11d7-9e83-815888ab59a7
```

Example 6.2.9. (Removing a given entry from LRC given the SFN)

```
$ GUID=`edg-lrc -i guidForPfn -h rlscms.cern.ch -p 7777 --vo cms
srm://cmslcgse01.cern.ch/flatfiles/lcg/cms/mu03_MB/cmkin/mu03_MB_77400311.n
tpl`
$ edg-lrc -i removePfn -h rlscms.cern.ch -p 7777 --vo cms $GUID
srm://cmslcgse01.cern.ch/flatfiles/lcg/cms/mu03_MB/cmkin/mu03_MB_77400311.n
tpl
```

Example 6.2.10. (Adding a new SFN to a given GUID in LRC)

```
$ edg-lrc -i addMapping -h rlscms.cern.ch -p 7777 --vo=cms \
guid:00329.ntpl-1059676658-724-7425 \
srm://grid015.pd.infn.it/shared/lcg/cms/mu03_MB/cmkin/mu03_MB_77400329.ntpl
```

7. INFORMATION SYSTEM

In the following section we give example on how to interrogate the Information System in LCG-1 Grid, in particular we discuss the information that can be obtained by the different servers: the local GRISes, the site GIISes, the global BDIIs. For a list of GLUE Schema ObjectClass and attributes, check APPENDIX B.

7.1. THE LOCAL GRIS

As explained before, the local GRISes running on Computing Elements and Storage Elements at the different sites report information on the characteristics and status of the services. They give both static and dynamic information.

In order to interrogate the GRIS on a specific Grid Element, one has to know the hostname of the Grid Element and the TCP port where the GRIS run. Such port is always 2135. One can then use the command:

```
$ ldapsearch -x -h <hostname> -p 2135 -b "mds-vo-name=local, o=grid"
```

Example 7.1.1. (Interrogating the GRIS on a Computing Element)

```
$ ldapsearch -x -h lxshare0235.cern.ch -p 2135 -b "mds-vo-name=local, o=grid"
```

```
version: 2
```

```
#
```

```
# filter: (objectclass=*)
```

```
# requesting: ALL
```

```
#
```

```
# lxshare0235.cern.ch/siteinfo, local, grid
```

```
dn: in=lxshare0235.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
```

```
objectClass: SiteInfo
```

```
objectClass: DataGridTop
```

```
objectClass: DynamicObject
```

```
siteName: LCGCERTTB1
```

```
sysAdminContact: "David.Smith@cern.ch, Marco.Serra@cern.ch,  
Di.Qing@cern.ch, Louis.Poncet@cern.ch"
```

```
userSupportContact: "David.Smith@cern.ch, Marco.Serra@cern.ch,  
Di.Qing@cern.ch, Louis.Poncet@cern.ch"
```

```
siteSecurityContact: "David.Smith@cern.ch, Marco.Serra@cern.ch,  
Di.Qing@cern.ch, Louis.Poncet@cern.ch"
```

```
dataGridVersion: t20030717_1431
```

```
installationDate: 20030717153000Z
```

```
# lxshare0235.cern.ch:2119/jobmanager-pbs-infinite, local, grid
```

```
dn: GlueCEUniqueID=lxshare0235.cern.ch:2119/jobmanager-pbs-infinite, mds-vo-name=local, o=grid
objectClass: GlueCETop
objectClass: GlueCE
objectClass: GlueSchemaVersion
objectClass: GlueCEAccessControlBase
objectClass: GlueCEInfo
objectClass: GlueCEPolicy
objectClass: GlueCEState
objectClass: GlueInformationService
objectClass: GlueKey
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCEHostingCluster: lxshare0235.cern.ch
GlueCEName: infinite
GlueCEUniqueID: lxshare0235.cern.ch:2119/jobmanager-pbs-infinite
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: lxshare0235.cern.ch
GlueCEInfoLRMSType: pbs
GlueCEInfoLRMSVersion: OpenPBS_2.4
GlueCEInfoTotalCPUs: 60
GlueCEStateEstimatedResponseTime: 0
GlueCEStateFreeCPUs: 60
GlueCEStateRunningJobs: 0
GlueCEStateStatus: Production
GlueCEStateTotalJobs: 0
GlueCEStateWaitingJobs: 0
GlueCEStateWorstResponseTime: 0
GlueCEPolicyMaxCPUTime: 172800
GlueCEPolicyMaxRunningJobs: 99999
GlueCEPolicyMaxTotalJobs: 999999
GlueCEPolicyMaxWallClockTime: 691200
GlueCEPolicyPriority: 1
GlueCEAccessControlBaseRule: VO:alice
GlueCEAccessControlBaseRule: VO:atlas
GlueCEAccessControlBaseRule: VO:cms
GlueCEAccessControlBaseRule: VO:lhcb
GlueCEAccessControlBaseRule: VO:wpsix
GlueCEAccessControlBaseRule: VO:iteam
GlueCEAccessControlBaseRule: VO:lcg
GlueCEAccessControlBaseRule: /C=AT/O=UIBK/OU=HEPG/CN=Reinhard
Bischof/Email=reinhard.bischof@uibk.ac.at
```

```
GlueCEAccessControlBaseRule: /C=CA/O=Grid/OU=phys.ualberta.ca/CN=Bryan L
Caron
GlueCEAccessControlBaseRule:/C=CN/O=NJU/OU=PD/CN=JialunPing/Email=jlping@pi
ne.njnu.edu.cn
[...]
GlueForeignKey: GlueClusterUniqueID=lxshare0235.cern.ch
GlueInformationServiceURL: ldap://lxshare0235.cern.ch:2135/mds-vo-
name=local, o=grid
[...]
```

In order to restrict the search to a specific Object Class and a specific attribute, one can specify the Object Class and the attribute in the query as in the following example. In the Appendix B it is possible to find a description of all Object Classes and their attributes to optimize the ldap search command.

Example 7.1.2. (Getting information about the site name from the GRIS on a Computing Element)

```
$ ldapsearch -x -h lxshare0235.cern.ch -p 2135 -b "mds-vo-name=local,
o=grid" 'objectclass=SiteInfo' siteName
version: 2

#
# filter: objectclass=SiteInfo
# requesting: siteName
#

# lxshare0235.cern.ch/siteinfo, local, grid
dn: in=lxshare0235.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
siteName: LCGCERTTB1

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1

$ ldapsearch -LLL -x -h lxshare0235.cern.ch -p 2135 -b "mds-vo-name=local,
o=grid" 'objectclass=SiteInfo' siteName
dn: in=lxshare0235.cern.ch/siteinfo,Mds-Vo-name=local,o=grid
siteName: LCGCERTTB1
```

7.2. THE SITE GIIS

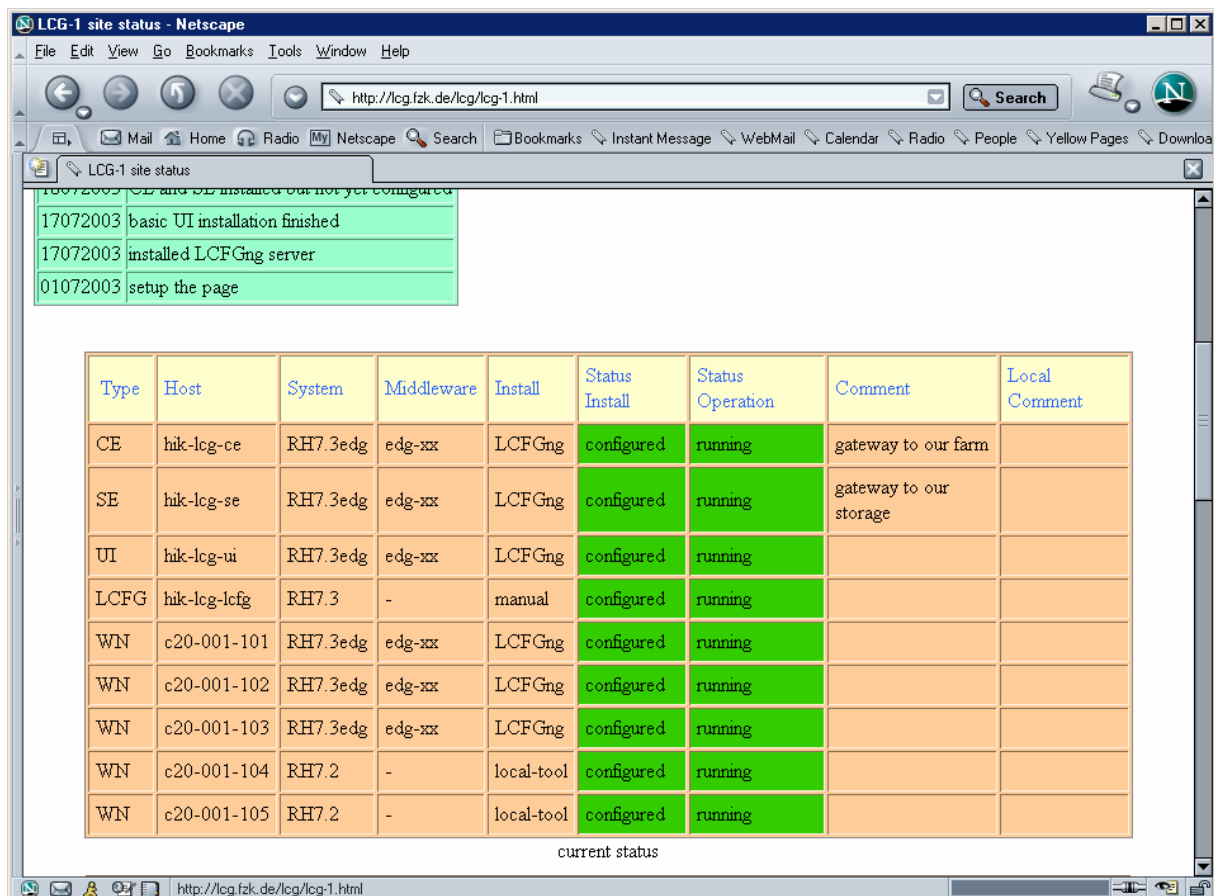
At each site a site GIIS collects information about **all resources** present at a site.

For a list of all sites and all resources present, please refer to:

`http://cern.ch/grid-deployment/cgi-bin/index.cgi?var=lcglStatus`

In order to interrogate such GIISes a different base name must be used. The base name can be obtained by the site name published by all sites, removing all "-" characters. So, for instance, for FZK, the site name is `FZK-LCG-1` and the mds base name is `mds-vo-name=fzklcg1, o=grid`.

Usually a site GIIS runs on the Computing Element on port 2135. In order to interrogate the site GIIS for FZK, one needs to find out the name of the Computing Element running the GIIS. This can be found on the WEB page reporting the site status: `http://lcg.fzk.de/lcg/lcg-1.html`.



The screenshot shows a Netscape browser window titled "LCG-1 site status - Netscape". The address bar contains "http://lcg.fzk.de/lcg/lcg-1.html". The page content includes a log of events and a table of system components.

| Type | Host | System | Middleware | Install | Status Install | Status Operation | Comment | Local Comment |
|------|--------------|----------|------------|------------|----------------|------------------|------------------------|---------------|
| CE | hik-lcg-ce | RH7.3edg | edg-xxx | LCFGng | configured | running | gateway to our farm | |
| SE | hik-lcg-se | RH7.3edg | edg-xxx | LCFGng | configured | running | gateway to our storage | |
| UI | hik-lcg-ui | RH7.3edg | edg-xxx | LCFGng | configured | running | | |
| LCFG | hik-lcg-lcfg | RH7.3 | - | manual | configured | running | | |
| WN | c20-001-101 | RH7.3edg | edg-xxx | LCFGng | configured | running | | |
| WN | c20-001-102 | RH7.3edg | edg-xxx | LCFGng | configured | running | | |
| WN | c20-001-103 | RH7.3edg | edg-xxx | LCFGng | configured | running | | |
| WN | c20-001-104 | RH7.2 | - | local-tool | configured | running | | |
| WN | c20-001-105 | RH7.2 | - | local-tool | configured | running | | |

current status

Figure 7

As we can see in Figure 7, the CE name is hik-lcg-ce.fzk.de.

So, in order to interrogate the site GIIS, we can use the following example:

Example 7.2.1. (Interrogating the site GUIS)

```
$ldapsearch -x -h hik-lcg-ce.fzk.de -p 2135 -b "mds-vo-name=fzk-lcg1,o=grid"
version: 2

#
# filter: (objectclass=*)
# requesting: ALL
#

# hik-lcg-ce.fzk.de/siteinfo, fzk-lcg1, grid
dn: in=hik-lcg-ce.fzk.de/siteinfo,Mds-Vo-name=fzk-lcg1,o=grid
objectClass: SiteInfo
objectClass: DataGridTop
objectClass: DynamicObject
siteName: FZK-LCG-1
sysAdminContact: lcg-admin@listserv.fzk.de
userSupportContact: lcg-admin@listserv.fzk.de
siteSecurityContact: lcg-admin@listserv.fzk.de
dataGridVersion: v1_5_0alpha
installationDate: 20030717184700Z

# hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long, fzk-lcg1, grid
dn: GlueCEUniqueID=hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long, Mds-Vo-
name=fzk-lcg1,o=grid
objectClass: GlueCETop
objectClass: GlueCE
objectClass: GlueSchemaVersion
objectClass: GlueCEAccessControlBase
objectClass: GlueCEInfo
objectClass: GlueCEPolicy
objectClass: GlueCEState
objectClass: GlueInformationService
objectClass: GlueKey
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCEHostingCluster: hik-lcg-ce.fzk.de
GlueCEName: long
GlueCEUniqueID: hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long
GlueCEInfoGatekeeperPort: 2119
GlueCEInfoHostName: hik-lcg-ce.fzk.de
GlueCEInfoLRMSType: pbs
```

```
[...]
# hik-lcg-se.fzk.de, hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long, fzklcg1,
grid
dn: GlueCESEBindSEUniqueID=hik-lcg-se.fzk.de,
GlueCESEBindGroupCEUniqueID=hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long,
Mds-Vo-name=fzklcg1,o=grid
objectClass: GlueGeneralTop
objectClass: GlueSchemaVersion
objectClass: GlueCESEBind
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueCESEBindCEAccesspoint: /flatfiles/SE00
GlueCESEBindCEUniqueID: hik-lcg-ce.fzk.de:2119/jobmanager-pbs-long
GlueCESEBindSEUniqueID: hik-lcg-se.fzk.de
[...]
# hik-lcg-se.fzk.de, fzklcg1, grid
dn: GlueSEUniqueID=hik-lcg-se.fzk.de,Mds-Vo-name=fzklcg1,o=grid
objectClass: GlueSETop
objectClass: GlueSE
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueSEUniqueID: hik-lcg-se.fzk.de
GlueSEName: FZK-LCG-1:edg-se
GlueForeignKey: GlueSLUniqueID=hik-lcg-se.fzk.de
GlueSEPort: 8080
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
GlueInformationServiceURL: ldap://hik-lcg-se.fzk.de:2135/Mds-Vo-
name=local,o=grid
[...]
# hik-lcg-se.fzk.de, fzklcg1, grid
dn: GlueSLUniqueID=hik-lcg-se.fzk.de,Mds-Vo-name=fzklcg1,o=grid
objectClass: GlueSLTop
objectClass: GlueSL
objectClass: GlueInformationService
objectClass: GlueKey
objectClass: GlueSchemaVersion
objectClass: GlueSLArchitecture
[...]
```

7.3. THE BDII

Each site running a Resource Broker runs as well a BDII that collects all information coming from the Regional GIISes and stores them in a permanent database. In order to find out the location of the BDII you can consult the WEB page of the LCG-1 site status as done for the site GIISes.

The BDII can be interrogated using the standard mds base: `mds-vo-name=local, o=grid`.

The BDII run on port 2170.

Example 7.3.1. (Interrogating a BDII)

```
$ ldapsearch -x -LLL -h lxshare0222.cern.ch -p 2170 -b "mds-vo-
name=local,o=grid" 'objectclass=GlueCESEBind' GlueCESEBindCEUniqueID
GlueCESEBindSEUniqueID
```

```
dn: GlueCESEBindSEUniqueID=grid100.kfki.hu,
GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119/jobmanager-pbs-infinite,
Mds-Vo-name=budapestlcg1, Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-infinite
GlueCESEBindSEUniqueID: grid100.kfki.hu
```

```
dn: GlueCESEBindSEUniqueID=grid100.kfki.hu,
GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119/jobmanager-pbs-long, Mds-
Vo-name=budapestlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-long
GlueCESEBindSEUniqueID: grid100.kfki.hu
```

```
dn: GlueCESEBindSEUniqueID=grid100.kfki.hu,
GlueCESEBindGroupCEUniqueID=grid109.kfki.hu:2119/jobmanager-pbs-short, Mds-
Vo-name=budapestlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: grid109.kfki.hu:2119/jobmanager-pbs-short
GlueCESEBindSEUniqueID: grid100.kfki.hu
```

```
dn: GlueCESEBindSEUniqueID=adc0021.cern.ch,
GlueCESEBindGroupCEUniqueID=adc0015.cern.ch:2119/jobmanager-lcgpbs-
infinite, Mds-Vo-name=cernlcg1,Mds-Vo-name=lcgeast,Mds-Vo-name=local,o=grid
GlueCESEBindCEUniqueID: adc0015.cern.ch:2119/jobmanager-lcgpbs-infinite
```

[...]

8. APPENDIX A

8.1. THE GRID MIDDLEWARE

The Grid Middleware deployed in the LCG-1 service is reported below.

The operating system for the computing elements is Linux Red Hat 7.3, mainly running on IA32 computers.

The LCG-1 Middleware layer uses components from EDT 1.1 (DataTag), EDG 2.0 (DataGrid) and VDT 1.1.8 (Virtual Data Toolkit). In the following we list the components from these packages/suites, which are actually used in LCG-1:

- EDG 2.0
 - EDG-WMS Workload Management System
 - Data Management System
 - EDG-RM (Replica Manager)
 - Including the EDG-ROS (Replica Optimisation Service)
 - EDG-RLS (Replica Location Service)
 - Including the EDG-RMC (Replica Metadata Catalog) and the EDG-LRC (Local Replica Catalog)
 - Fabric Management
 - EDG WP4 tools/procedures (LCFG, LCFG-Lite or manual procedures)
 - LCAS/LCMAPS (Local Centre Auth. System and Local Mapping)
 - Virtual Organization Management
 - EDG infrastructure and procedures
 - Information service
 - Information index BDII
- EDT 1.1
 - Monitoring system
 - Grid-ICE
 - Glue Schema LCG-EDT 1.2
- VDT 1.1.8
 - Core components:
 - Globus 2.2.4
 - Condor 6.4.7
 - Condor-G

-
- ClassAds
 - Information Service
 - Globus MDS
 - GRIS (Grid Resource Information Service)
 - GIIS (Grid Index Information Service)

9. APPENDIX B

9.1. THE GLUE SCHEMA

In this appendix, all the attributes defined in the information schema and their object classes are defined. Some of the attributes may actually be empty, even if they are defined in the schema.

9.1.1. Attributes for the Computing Element

- **CE** (objectclass `GlueCE`)
 - `GlueCEUniqueID`: unique identifier for the CE
 - `GlueCEName`: human-readable name of the service
- **Info** (objectclass `GlueCEInfo`)
 - `GlueCEInfoLRMSType`: name of the local batch system
 - `GlueCEInfoLRMSVersion`: version of the local batch system
 - `GlueCEInfoGRAMVersion`: version of GRAM
 - `GlueCEInfoHostName`: fully qualified name of the host where the gatekeeper runs
 - `GlueCEInfoGateKeeperPort`: port number for the gatekeeper
 - `GlueCEInfoTotalCPUs`: number of CPUs in the cluster associated to the CE
- **Policy** (objectclass `GlueCEPolicy`)
 - `GlueCEPolicyMaxWallClockTime`: maximum wall clock time available to jobs submitted to the CE
 - `GlueCEPolicyMaxCPUtime`: maximum CPU time available to jobs submitted to the CE
 - `GlueCEPolicyMaxTotalJobs`: maximum allowed total number of jobs in the queue
 - `GlueCEPolicyMaxRunningJobs`: maximum allowed number of running jobs in the queue
 - `GlueCEPolicyPriority`: information about the service priority
- **State** (objectclass `GlueCEState`)
 - `GlueCEStateRunningJobs`: number of running jobs
 - `GlueCEStateWaitingJobs`: number of jobs not running
 - `GlueCEStateTotalJobs`: total number of jobs (running + waiting)

-
- `GlueCEStateStatus`: queue status: queueing (jobs are accepted but not run), production (jobs are accepted and run), closed (jobs are neither accepted nor run), draining (jobs are not accepted but those in the queue are run)
 - `GlueCEStateWorstResponseTime`: worst possible time between the submission of a job and the start of its execution
 - `GlueCEStateEstimatedResponseTime`: estimated time between the submission of a job and the start of its execution
 - `GlueCEStateFreeCPUs`: number of CPUs available to the scheduler
 - **Job** (currently not filled, the Logging and Bookkeeping service can provide this information) (objectclass `GlueCEJob`)
 - `GlueCEJobLocalOwner`: local user name of the job's owner
 - `GlueCEJobGlobalOwner`: GSI subject of the real job's owner
 - `GlueCEJobLocalID`: local job identifier
 - `GlueCEJobGlobalId`: global job identifier
 - `GlueCEJobGlueCEJobStatus`: job status: SUBMITTED, WAITING, READY, SCHEDULED, RUNNING, ABORTED, DONE, CLEARED, CHECKPOINTED
 - `GlueCEJobSchedulerSpecific`: any scheduler specific information
 - **Access control** (objectclass `GlueCEAccessControlBase`)
 - `GlueCEAccessControlBaseRule`: a rule defining any access restrictions to the CE. Current semantic: VO = a VO name, DENY = an X.509 user subject
 - **Cluster** (objectclass `GlueCluster`)
 - `GlueClusterUniqueID`: unique identifier for the cluster
 - `GlueClusterName`: human-readable name of the cluster
 - **Subcluster** (objectclass `GlueSubCluster`)
 - `GlueSubClusterUniqueID`: unique identifier for the subcluster
 - `GlueSubClusterName`: human-readable name of the subcluster
 - **Host** (objectclass `GlueHost`)
 - `GlueHostUniqueId`: unique identifier for the host
 - `GlueHostName`: human-readable name of the host
 - **Architecture** (objectclass `GlueHostArchitecture`)
 - `GlueHostArchitecturePlatformType`: platform description

-
- GlueHostArchitectureSMPSize: number of CPUs
 - **Operating system** (objectclass GlueHostOperatingSystem)
 - GlueHostOperatingSystemOSName: OS name
 - GlueHostOperatingSystemOSRelease: OS release
 - GlueHostOperatingSystemOSVersion: OS or kernel version
 - **Benchmark** (objectclass GlueHostBenchmark)
 - GlueHostBenchmarkSI00: SpecInt2000 benchmark
 - GlueHostBenchmarkSF00: SpecFloat2000 benchmark
 - **Application software** (objectclass GlueHostApplicationSoftware)
 - GlueHostApplicationSoftwareRunTimeEnvironment: list of software installed on this host
 - **Processor** (objectclass GlueHostProcessor)
 - GlueHostProcessorVendor: name of the CPU vendor
 - GlueHostProcessorModel: name of the CPU model
 - GlueHostProcessorVersion: version of the CPU
 - GlueHostProcessorOtherProcessorDescription: other description for the CPU
 - GlueHostProcessorClockSpeed: clock speed of the CPU
 - GlueHostProcessorInstructionSet: name of the instruction set architecture of the CPU
 - GlueHostProcessorGlueHostProcessorFeatures: list of optional features of the CPU
 - GlueHostProcessorCacheL1: size of the unified L1 cache
 - GlueHostProcessorCacheL1I: size of the instruction L1 cache
 - GlueHostProcessorCacheL1D: size of the data L1 cache
 - GlueHostProcessorCacheL2: size of the unified L2 cache
 - **Main memory** (objectclass GlueHostMainMemory)
 - GlueHostMainMemoryRAMSize: physical RAM
 - GlueHostMainMemoryRAMAvailable: unallocated RAM
 - GlueHostMainMemoryVirtualSize: size of the configured virtual memory
 - GlueHostMainMemoryVirtualAvailable: available virtual memory
 - **Network adapter** (objectclass GlueHostNetworkAdapter)

-
- GlueHostNetworkAdapterName: name of the network card
 - GlueHostNetworkAdapterIPAddress: IP address of the network card
 - GlueHostNetworkAdapterMTU: the MTU size for the LAN to which the network card is attached
 - GlueHostNetworkAdapterOutboundIP: permission for outbound connectivity
 - GlueHostNetworkAdapterInboundIP: permission for inbound connectivity
 - **Processor load** (objectclass GlueHostProcessorLoad)
 - GlueHostProcessorLoadLast1Min: one-minute average processor availability for a single node
 - GlueHostProcessorLoadLast5Min: 5-minute average processor availability for a single node
 - GlueHostProcessorLoadLast15Min: 15-minute average processor availability for a single node
 - **SMP load** (objectclass GlueHostSMPLoad)
 - GlueHostSMPLoadLast1Min: one-minute average processor availability for a single node
 - GlueHostSMPLoadLast5Min: 5-minute average processor availability for a single node
 - GlueHostSMPLoadLast15Min: 15-minute average processor availability for a single node
 - **Storage device** (objectclass GlueHostStorageDevice)
 - GlueHostStorageDeviceName: name of the storage device
 - GlueHostStorageDeviceType: storage device type
 - GlueHostStorageDeviceTransferRate: maximum transfer rate for the device
 - GlueHostStorageDeviceSize: Size of the device
 - GlueHostStorageDeviceAvailableSpace: amount of free space
 - **Local file system** (objectclass GlueHostLocalFileSystem)
 - GlueHostLocalFileSystemRoot: path name or other information defining the root of the file system
 - GlueHostLocalFileSystemSize: size of the file system in bytes
 - GlueHostLocalFileSystemAvailableSpace: amount of free space in bytes

-
- `GlueHostLocalFileSystemReadOnly`: true if the file system is read-only
 - `GlueHostLocalFileSystemType`: file system type
 - `GlueHostLocalFileSystemName`: the name for the file system
 - `GlueHostLocalFileSystemClient`: host unique id of clients allowed to remotely access this file system
 - **Remote file system** (objectclass `GlueHostRemoteFileSystem`)
 - `GlueHostLRemoteFileSystemRoot`: path name or other information defining the root of the file system
 - `GlueHostRemoteFileSystemSize`: size of the file system in bytes
 - `GlueHostRemoteFileSystemAvailableSpace`: amount of free space in bytes
 - `GlueHostRemoteFileSystemReadOnly`: true if the file system is read-only
 - `GlueHostRemoteFileSystemType`: file system type
 - `GlueHostRemoteFileSystemName`: the name for the file system
 - `GlueHostRemoteFileSystemServer`: host unique id of the server which provides access to the file system
 - **File** (objectclass `GlueHostFile`)
 - `GlueHostFileName`: name for the file
 - `GlueHostFileSize`: file size in bytes
 - `GlueHostFileCreationDate`: file creation date and time
 - `GlueHostFileLastModified`: date and time of the last modification of the file
 - `GlueHostFileLastAccessed`: date and time of the last access to the file
 - `GlueHostFileLatency`: time taken to access the file in seconds
 - `GlueHostFileLifeTime`: time for which the file will stay on the storage device
 - `GlueHostFileOwner`: name of the owner of the file

9.1.2. Attributes for the Storage Element

- **Storage Service** (objectclass `GlueSE`)
 - `GlueSEUniqueId`: unique identifier of the storage service (URI)
 - `GlueSEName`: human-readable name for the service
 - `GlueSEPort`: port number that the service listens

-
- `GlueSEHostingSL`: unique identifier of the storage library hosting the service
 - **Storage Service State** (objectclass `GlueSEState`)
 - `GlueSEStateCurrentIOLoad`: system load (for example, number of files in the queue)
 - **Storage Service Access Protocol** (objectclass `GlueSEAccessProtocol`)
 - `GlueSEAccessProtocolType`: protocol type to access or transfer files
 - `GlueSEAccessProtocolPort`: port number for the protocol
 - `GlueSEAccessProtocolVersion`: protocol version
 - `GlueSEAccessProtocolAccessTime`: time to access a file using this protocol
 - `GlueSEAccessProtocolSupportedSecurity`: security features supported by the protocol
 - **Storage Library** (objectclass `GlueSL`)
 - `GlueSLName`: human-readable name of the storage library
 - `GlueSLUniqueId`: unique identifier of the machine providing the storage service
 - `GlueSLService`: unique identifier for the provided storage service
 - **Local File system** (objectclass `GlueSLLocalFileSystem`)
 - `GlueSLLocalFileSystemRoot`: path name (or other information) defining the root of the file system
 - `GlueSLLocalFileSystemName`: name of the file system
 - `GlueSLLocalFileSystemType`: file system type (e.g. NFS, AFS, etc.)
 - `GlueSLLocalFileSystemReadOnly`: true is the file system is read-only
 - `GlueSLLocalFileSystemSize`: total space assigned to this file system
 - `GlueSLLocalFileSystemAvailableSpace`: total free space in this file system
 - `GlueSLLocalFileSystemClient`: unique identifiers of clients allowed to access the file system remotely
 - `GlueSLLocalFileSystemServer`: unique identifier of the server exporting this file system (only for remote file systems)
 - **Remote File system** (objectclass `GlueSLRemoteFileSystem`)
 - `GlueSLRemoteFileSystemRoot`: path name (or other information) defining the root of the file system
 - `GlueSLRemoteFileSystemName`: name of the file system

-
- `GlueSLRemoteFileSystemType`: file system type (e.g. NFS, AFS, etc.)
 - `GlueSLRemoteFileSystemReadOnly`: true is the file system is read-only
 - `GlueSLRemoteFileSystemSize`: total space assigned to this file system
 - `GlueSLRemoteFileSystemAvailableSpace`: total free space in this file system
 - `GlueSLRemoteFileSystemServer`: unique identifier of the server exporting this file system
 - **File Information** (objectclass `GlueSLFile`)
 - `GlueSLFileName`: file name
 - `GlueSLFileSize`: file size
 - `GlueSLFileCreationDate`: file creation date and time
 - `GlueSLFileLastModified`: date and time of the last modification of the file
 - `GlueSLFileLastAccessed`: date and time of the last access to the file
 - `GlueSLFileLatency`: time needed to access the file
 - `GlueSLFileLifeTime`: file lifetime
 - `GlueSLFilePath`: file path
 - **Directory Information** (objectclass `GlueSLDirectory`)
 - `GlueSLDirectoryName`: directory name
 - `GlueSLDirectorySize`: directory size
 - `GlueSLDirectoryCreationDate`: directory creation date and time
 - `GlueSLDirectoryLastModified`: date and time of the last modification of the directory
 - `GlueSLDirectoryLastAccessed`: date and time of the last access to the directory
 - `GlueSLDirectoryLatency`: time needed to access the directory
 - `GlueSLDirectoryLifeTime`: directory lifetime
 - `GlueSLDirectoryPath`: directory path
 - **Architecture** (objectclass `GlueSLDirectory`)
 - `GlueSLDirectoryType`: type of storage hardware (i.e. disk, RAID array, tape library, etc.)
 - **Performance** (objectclass `GlueSLPerformance`)
 - `GlueSLPerformanceMaxIOCapacity`: maximum bandwidth between the service and the network

-
- **Storage Space** (objectclass `GlueSA`)
 - `GlueSARoot`: pathname of the directory containing the files of the storage space
 - **Policy** (objectclass `GlueSAPolicy`)
 - `GlueSAPolicyMaxFileSize`: maximum file size
 - `GlueSAPolicyMinFileSize`: minimum file size
 - `GlueSAPolicyMaxData`: maximum allowed amount of data that a single job can store
 - `GlueSAPolicyMaxNumFiles`: maximum allowed number of files that a single job can store
 - `GlueSAPolicyMaxPinDuration`: maximum allowed lifetime for non-permanent files
 - `GlueSAPolicyQuota`: total available space
 - `GlueSAPolicyFileLifeTime`: lifetime policy for the contained files
 - **Access Control Base** (objectclass `GlueSAAccessControlBase`)
 - `GlueSAAccessControlBase Rule`: list of the access control rules
 - **State** (objectclass `GlueSAState`)
 - `GlueSAStateAvailableSpace`: total space available in the storage space
 - `GlueSAStateUsedSpace`: used space in the storage space

10. APPENDIX C

10.1. TROUBLESHOOTING

10.1.1. Job Submission

10.1.2. Data Management

10.1.3. Information System

11. APPENDIX D

11.1. JOB STATUS DEFINITION

As already mentioned in chapter 5, a job can find itself in one of several possible states, the definition of which is given in this table.

| Status | Definition |
|-----------|---|
| Submitted | The job has been submitted by the user but not yet processed by the Network Server |
| Waiting | The job has been accepted by the Network Server but not yet processed by the Workload Manager |
| Ready | The job has been assigned to a Computing Element but not yet transferred to it |
| Scheduled | The job is waiting in the Computing Element's queue |
| Running | The job is running |
| Done | The job has finished |
| Aborted | The job has been aborted by the WMS (e.g. because it was too long, or the proxy certificated expired, etc.) |
| Cancelled | The job has been cancelled by the user |
| Cleared | The output sandbox has been transferred to the User Interface |

Only some transitions between states are allowed. These transitions are depicted in the following figure.

