Proposal/Contract: 031874

CYCLOPS

CYber-Infrastructure for CiviL protection Operative ProcedureS

EGEE cookbook: a guide for Civil Protection Grid users

Reference : **CYCLOPS-WP02-D7-INFN**

Due date of deliverable: 30 / 11 / 2006

Start date of the project: **JUNE 1st 2006** Duration: **24 months**

Partner: **INFN**

Editor(s): **Stefano Dal Pra**

Issue: 01 **Revision**: **01**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Deliverable Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

DOCUMENT HISTORY

Issue.revision	Date	Description of change	Author	Affiliation
1.0	20 May 2007	Creation	Stefano Dal Pra	INFN
1.1	25 May 2007	Some Refinements	Marco Verlato	INFN

DISTRIBUTION LIST

PARTNERS:

DPC	INFN	IMAA	DDSC	CP-CH	TEI-CR	SNBPC
EXTERNAL EXPERTS						
OTHERS:						
CYCLOPS WEB SITE	http://www.cyclops-project.eu					

TABLES OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The purpose of this document is to provide a quick list of steps one needs to perform in order to work with the GRID infrastructure and aims at providing a minimal insight on different kind of activities which can be performed. After reading this document You should be able to:

- Subscribe to the cyclops Virtual Organization (cyclops VO)
- Create and submit test jobs to the GRID.
- Check the status of your submitted jobs from the GRID Viewpoint.
- Manage data transfers to/from the GRID resources.
- define jobs related each other through dependencies (MPI, Collections, DAG jdl)

This document is part of CYCLOPS as deliverable D7.

1.2 APPLICATION AREA

The main target audience of this document are the CYCLOPS partners, but all stakeholders in the CYCLOPS project are potential recipients of EGEE dissemination and training events.

1.3 REFERENCES

[R1] http://www.cyclops-project.eu/ CYCLOPS Technical Annex I

1.4 DOCUMENT AMENDMENT PROCEDURE

Requests to amend this document must be sent in the first place to the main author (Stefano Dal Pra, email stefano.dalpra@pd.infn.it).

1.5 TERMINOLOGY

CYCLOPS	Cyber-infrastructure for CiviL protection Operative Procedures
TA	Technical Annex
WP	Work Package
GILDA	Grid INFN Laboratory for Dissemination Activities
GMES	Global Monitoring for Environment and Security
EGEE	Enabling Grid in E-sciencE
CA	Certification Authority
RA	Registration Authority
RB	Resource Broker
LRMS	Local Resource Management System (batch system)
WMS	Workload Manager Service
VO	Virtual Organization
VOMS	Virtual Organization Membership Service

2 PRELIMINARY STEPS

2.1 ACCEPTING THE AUP

Read and accept the "Grid Acceptable Use Policy" which states terms of use for the European GRID infrastructure. You can download this document from here: https://edms.cern.ch/file/428036/3/Grid_AUP.pdf. Later on, when registering to the cyclops VO you'll declare that you are aware of and that you accept these terms of use.

2.2 OBTAINING A PERSONAL DIGITAL CERTIFICATE

Certificates are extensively used in Grid environment for authentication and authorization purposes. Digital certificates used in European Grids are released by Certificate Authorities which are members of EUGridPMA Organization (see http://eugridpma.org/). Also, certificate requests received from CA must be previously validated from the competent and trusted Registration Authority (RA) who is in charge of assuring the identity of the person issuing a certificate request. This means that you meet face-to-face the competent RA for your CA, and you both get sure of each other's identity.

2.2.1 How to get a personal certificate

First of all, before you can get access to the Cyclops-GRID infrastructure, you must own a Personal Certificate, issued by one of the Certification Authorities (CAs) trusted by the project. The way you get a personal certificate may vary depending on your country and/or local organization. Here are example steps:

- Go to http://eugridpma.org/members/worldmap/ where you can identify the right CA for your country. We assume in this example to have selected the Italian CA: http://security.fi.infn.it/CA/en/ .
- Follow the "Personal Certificate request" link:

  https://security.fi.infn.it/CA/en/mgt/restricted/ucert.php
- Ask your local manager for details. As a general rule, be sure to use the very same browser (in the same computer) in every required step you'll follow to ask for the certificate. Also pay attention not to forget any password or pass-phrase you should be asked of.
- Certificates used in the Grid environment need to be in "PEM" format, but our own may probably be in PK12 format. You can convert the format from PK12 to PEM with the following commands:

```
openssl pkcs12 -nocerts -in my_cert.p12 -out userkey.pem

openssl pkcs12 -clcerts -nokeys -in my_cert.p12 -out usercert.pem
```

2.2.2 Certificate and key

After getting your certificate you should end up with having two files:

- A file with your private key: userkey.pem

  this one has been generated just before requesting the certificate and no one but you should see it. It should be readable by you only.
- A file with your Personal Certificate: usercert.pem

  this file has been given to you by the CA and may be world readable.

2.3 LOAD THE CERTIFICATE

Install your Personal Certificate in your web browser. This is needed for any authenticated web operation you'll perform from now on, such as the next one.

2.4 REGISTERING TO THE VO

With your personal certificate installed in your browser, you can perform the cyclops VO registration request. Open with your browser the following url: https://voms2.cnaf.infn.it:8443/voms/cyclops/webui/request/user/create. You'll notice a request form with primary fields (First Name, Family Name) already filled in. Complete the form and submit your subscription request by clicking the AUP acceptance button. Your request will be emailed to the VO admin which will consider it for acceptance. You should typically receive three emails, which subjects are:

1. Email address confirmation for VO cyclops
2. Accepted email confirmation for VO cyclops

3. Welcome to the cyclops VO!

   After receiving the third one you're ready to submit your first jobs to the cyclops VO.

3 JOB SUBMISSION QUICK TOUR

3.1 INTRODUCTION

This chapter illustrates an almost minimal set of things needed to submit a job to the grid, to check its status and to retrieve its output results, in order to get acquainted with basic operations on the grid.

Two things are needed to access the grid: a User Interface (the set of commands to interact with grid) and your personal certificate and private key (needed for authentication and authorization purposes).

A few words must be said about commands and how they relate to Grid elements. There are three main command families appeared in time order:

- edg-* Grid general-purpose commands

  This is the first developed command set.

- lcg-* Grid utilities commands.
- glite-* Grid general-purpose commands and utilities. They access the WMS (Workload Management System) through the NS (Network Server).
- glite-wms-* Grid commands to access the WMS through the WMProxy.

If you would like to know exactly how many and which commands are into the, say, glite-wms family, just type it and the press twice the TAB key: you'll see all possible command completions.

We are going to illustrate in the examples below the latest command set glite-wms-*. Eventually we could make use of a few commands from other groups. The glite commands family has been developed in order to take advantage of new Grid technologies while remaining compatible when new elements or services aren't available.

Production Grid makes use of lcg based Resource Brokers. A RB is a Grid element which is responsible to find suitable Grid resources for our jobs to run. A key feature for a RB relies on a fair queuing model, which is provided, in glite based RB, by the “Workload Management Service” (WMS). gLite based RB (commonly called glite-WMS or simply WMS) has been extensively tested in pre-production Grids and is going to be used in production Grid, thus we are in a transitional phase, at the time of this writing. For this reason we'll focus our examples mainly on glite and glite-wms commands.

3.2 ACCESS YOUR USER INTERFACE

The User Interface (UI) is the front end one uses to interact with the Grid, i.e. to submit jobs, store and retrieve files and so on. It mainly consists of a bunch of GNU/Linux text command line programs in a pre-configured environment. There are four kinds of UI, simplest and easiest of which is the Plug'n Play UI.

See http://grid-it.cnaf.infn.it/index.php?userinterface&type=1 for updated download and install instructions.

3.2.1 Enable the UI

Whichever UI you choose, your next step is to create a .globus directory which will contain your personal certificate and your private key. Doing this is a matter of issuing these or similar commands:

```
mkdir .globus

mv userkey.pem .globus

mv usercert.pem .globus

chmod 700 .globus

cd .globus

chmod 400 userkey.pem
```

chmod 644 usercert.pem

3.3 FIRST STEPS WITH UI

We are finally ready to work with the UI. Our first commands are used to obtain and check for a proxy certificate. This is a sort of “temporary personal certificate” (it expires by default after 12 hours and options are available to extend its lifetime) which is valid to access Grid resources depending on your privileges. Furthermore we would get a “delegated proxy” which is needed when issuing glite-wms commands. Since we are registered on the cyclops VO, with the role assigned to us from the cyclops VO manager, we'll be able to only access cyclops related resources, with usage rights restricted by our role's privileges. We can consider the proxy certificate as our “temporary pass” for the Grid.

3.3.1 Creating a Proxy-certificate

Assuming you are logged into a UI, issue following command:

```
[sdp@glite-ui]$ voms-proxy-init --voms cyclops
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra
Enter GRID pass phrase:
Creating temporary proxy ..... Done
Contacting  voms2.cnaf.infn.it:15011 [/C=IT/O=INFN/OU=Host/L=CNAF/CN=voms2.cnaf.infn.it] "cyclops" Done
Creating proxy ..... Done
Your proxy is valid until Wed Mar 21 02:19:45 2007
```

As you can see, you'll be asked for your private key's pass-phrase (should your private key be stolen, it couldn't be useful without knowing the pass-phrase). Also Notice how long your proxy lasts.

3.3.2 Verifying the proxy-certificate

Let's check about our proxy:

```
[sdp@glite-ui]$ voms-proxy-info --all
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u543
timeleft  : 11:59:45
VO        : cyclops
subject   : /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra
issuer    : /C=IT/O=INFN/OU=Host/L=CNAF/CN=voms2.cnaf.infn.it
attribute : /cyclops/Role=NULL/Capability=NULL
timeleft  : 12:03:40
```

Note the “attribute” field, telling you which VO's access the proxy enables you and with which privileges.

3.3.3 Delegating proxy

A new peculiarity of glite-WMS consists in a "delegation proxy" mechanism that simplifies authentication and authorization steps between Grid elements. This step may be implicitly done by adding a -a option to glite-wms commands (this family makes use of the WMproxy service, while glite-job commands use the older Network Server), but for the sake of clarity we explicitly issue the delegation command:

```
[sdp@glite-ui]$ glite-wms-job-delegate-proxy -d friendly_del_id -o my_del_ids.txt

Connecting to the service https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server

================== glite-wms-job-delegate-proxy Success ==================

Your proxy has been successfully delegated to the WMProxy:

https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server

with the delegation identifier: friendly_del_id

The DelegateProxy result  has been saved in the following file:

/home_local/sdalpra/my_del_ids.txt

==========================================================================
```

The -d switch let us specify the id-string so we can choose a friendly one; we could instead use the -a option, then we would get an automatically generated one. The optional -o switch specify a file into which our delegation ids will be saved.

3.3.4 Looking for resources

Now it is a good time to look for resources where we could consider submitting our stuff. This is an optional step, but it tell us at a glance what is available for us, as cyclops VO users. Note this is a lcg command.

```
[sdp@glite-ui]$ lcg-infosites  --vo cyclops all

valor del bdii: egee-bdii.cnaf.infn.it:2170
#CPU  Free  Total Jobs Running Waiting  ComputingElement
-------------------------------------------------------------
 4     0      0        0       0     gridce.ilc.cnr.it:2119/jobmanager-lcgpbs-grid
 10    1      9        0       9     gridit-ce-001.cnaf.infn.it:2119/jobmanager-lcgpbs-egee
 10    1      9        0       9     glite-ce-01.cnaf.infn.it:2119/blah-pbs-egee
 48    24     17       0       17    grid0.fe.infn.it:2119/jobmanager-lcgpbs-grid
 62    0      27       21      6     prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
 37    0      31       19      12    grid003.roma2.infn.it:2119/jobmanager-lcgpbs-grid
 15    4      2        1       1     gridce.sns.it:2119/jobmanager-lcgpbs-grid

Avail Space(Kb) Used Space(Kb)  Type    SEs
-------------------------------------------------------------
395252028          4874600      n.a    gridse.ilc.cnr.it
479999416       1329338248      n.a    grid007g.cnaf.infn.it
3937229840       456948720      n.a    prod-se-01.pd.infn.it
52670524         656381100      n.a    gridit002.pd.infn.it
3850000000       240000000      n.a    prod-se-02.pd.infn.it
67016288           4009492      n.a    gridse.sns.it
```

We discover that 7 Computing Elements (CEs) are available for job submission and the list of available Storage Elements is also given.

Strictly speaking a CE is identified by a string of the form: <hostname>:<port>/<queue>, so it is possible to have more than one CE on a single host. The <queue> part, in turn, follows this syntax: <GridGateway_type>-<LRMS_type>-<batch_queue_name>.

3.3.5 Helloworld.jdl

We are now ready to submit a test job. We define a simplest possible job using the "Job Description Language". Edit a text file named Helloworld.jdl with this content:

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "message.txt";
StdError = "stderror";
OutputSandbox = {"message.txt","stderror"};
```

Parameters meaning is almost self explanatory. You specify the executable you want to be run, its arguments, which files would contain runtime output messages and error messages. Finally you specify your "OutputSandbox" which is a sort of container with any output generated by your executable. Notice that you must know in advance name and number of files that your executable will produce.

3.3.6 May I submit it?

Not surprisingly, the Helloworld.jdl requires nothing particular to be executed, but real world applications may require to write complex jdl with complex executable needs, thus we cannot be sure in advance if any available grid resource can successfully run our job. Also, a failure may happen after lots of resources consumption, causing wasting of time and resources. Even more simply, our jdl may contain a syntax error and we would notice this only after having submitted it to the Grid. To prevent this, a specific command exists. It checks which grid resources are available to run a given jdl file. Note how we specify the "delegation proxy id" that we've seen in previous glite-wms command:

```
[sdp@glite-ui]$ glite-wms-job-list-match -d friendly_del_id  HelloWorld.jdl
Connecting to the service https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server
==========================================================================
            COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:
      *CEId*
 - glite-ce-01.cnaf.infn.it:2119/blah-pbs-egee
 - grid0.fe.infn.it:2119/jobmanager-lcgpbs-grid
 - gridce.sns.it:2119/jobmanager-lcgpbs-grid
 - grid003.roma2.infn.it:2119/jobmanager-lcgpbs-grid
 - gridce.ilc.cnr.it:2119/jobmanager-lcgpbs-grid
 - gridit-ce-001.cnaf.infn.it:2119/jobmanager-lcgpbs-egee
 - prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
==========================================================================
```

As we can see, all of the available cyclops queues can satisfy our job requirements. We can proceed and submit our job.

In this example we explicitly specified a "delegation proxy id", but we could "shortcut" this step by using instead the -a (automatic) switch. We'll use it in next examples, bypassing the need for glite-wms-job-delegate-proxy command.

Previous and older glite command (not making use of WMProxy) would be issued this way:

```
[sdp@glite-ui]$ glite-job-list-match HelloWorld.jdl
```

```
Selected Virtual Organisation name (from proxy certificate extension): cyclops
Connecting to host prod-wms-01.pd.infn.it, port 7772
**********************************************************************
                COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:
       *CEId*
 - glite-ce-01.cnaf.infn.it:2119/blah-pbs-egee
 - grid0.fe.infn.it:2119/jobmanager-lcgpbs-grid
 - gridce.sns.it:2119/jobmanager-lcgpbs-grid
 - grid003.roma2.infn.it:2119/jobmanager-lcgpbs-grid
 - gridce.ilc.cnr.it:2119/jobmanager-lcgpbs-grid
 - gridit-ce-001.cnaf.infn.it:2119/jobmanager-lcgpbs-egee
 - prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
**********************************************************************
```

without need for -d nor -a switch.

3.3.7 Let's submit it!

We are ready to submit our job. The simplest possible way is through glite-wms-job-submit -a <jobdesc.jdl>. Since we learnt from previous command which queues are available we could explicitly select one of them using the -r (resource) switch:

```
glite-wms-job-submit -a -r prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid HelloWorld.jdl
```

If we had a good reason to directly specify the queue we would skip the time needed to the Resource Broker (The actual grid service based on WMProxy or Network Server) to choose a good one for us, resulting in a reduced execution latency. On the other way, specifying a slower or crowded queue, we would loose any advantage and probably get worse results. Direct queue choice should be avoided, since it could lead to unwanted side-effects. Instruments to evaluate which queue would be better for us are available for experienced users and we'll make mention about a few of them; here we just submit our job the simple way. Below is an example.

```
[sdp@glite-ui]$ glite-wms-job-submit -a HelloWorld.jdl
Connecting to the service https://glite-rb-00.cnaf.infn.it:7443/glite_wms_wmproxy_server
====================== glite-wms-job-submit Success ======================
The job has been successfully submitted to the WMProxy
Your job identifier is:
https://glite-rb-00.cnaf.infn.it:9000/zjL3DRH4Eas2ud37jACN7A
==========================================================================
```

You have to take note of the job identifier assigned to your submission, since you'll need to specify it for any subsequent command related to this job, else you'll not be able to retrieve results when finished. This is especially important if you perform many submissions. You are encouraged to use the -o myjobids.txt for any job submission. This file will contain any jobId (one per line) for your submissions.

3.3.8 Where is my job?

After a while your job should have done. You can check for its status using glite-wms-job-status:

```
[sdp@glite-ui]$ glite-wms-job-status https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
**********************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
Current Status:     Done (Success)
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
Submitted:          Thu Mar 22 09:58:33 2007 CEST
**********************************************************
```

As you can see the job has successfully finished. Current status may be one of a few many values (Submitted, Scheduled, Running,..., Done). If the job is in a final state (i.e. it will progress no more) you should also see an Exit code (0 for success).

If you used the -o option when submitting the job you can provide -i myjobids.txt to glite-wms-job-status and you'll be asked for which job you are interested to (simply type <return> to see them all).

3.3.9 Cancelling a job

You may eventually realize that you do not need to wait for a submitted job to complete. You can cancel such job, thus preventing useless waste of Grid resources. The command to do this is simply glite-wms-job-cancel <job_id> :

```
[sdp@glite-ui]$ glite-wms-job-cancel https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
Are you sure you want to remove specified job(s) [y/n]y : y
Connecting to the service https://193.206.210.111:7443/glite_wms_wmproxy_server
============================= glite-wms-job-cancel Success =============================
The cancellation request has been successfully submitted for the following job(s):
- https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
========================================================================================
```

3.3.10 Retrieving results

Ok, now that the job has succesfully finished we want to retrieve its results:

```
[sdp@glite-ui]$ glite-wms-job-output --dir ./myjobresults\
https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
Retrieving files from host: prod-wms-01.pd.infn.it \
( for https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g )
************************************************************************************
                JOB GET OUTPUT OUTCOME
 Output sandbox files for the job:
 - https://prod-wms-01.pd.infn.it:9000/xtyIfxqTH5dnmYT_Yi4g2g
 have been successfully retrieved and stored in the directory:
 /home/sdp/myjobresults
************************************************************************************
```

3.3.11 Caveat

After this command the SandboxDir with job's results gets MOVED from the Grid side to your UI. Keep in mind that you cannot retrieve twice the results for the same job. After a glite-wms-job-output, Grid side resources can be freed. It is a good habit to retrieve SandBoxes for finished jobs, even thought they will be automatically purged when older than about a couple of weeks.

3.3.12 What Next?

Ok, this chapter was intended as a "quickest possible" tour to let you start with job submission. Real work can be performed just as depicted through this chapter, using a real jdl file. Of course special needs may (and should) arise during your Grid experience. You can learn more ways to use a command by simply issuing it without arguments, or providing the --help option: the command will show its on line help.

We are going now to very shortly describe main Grid elements. We will refer to them when describing more in depth operations. Also a few unexplained things seen in the above examples should become clearer.

4 GRID ELEMENTS AND COMPONENTS

4.1 COMPUTING ELEMENT (CE)

A Computing Element (CE), in Grid terminology, represents a set of computing resources in a site (a cluster, a farm) which takes in charge jobs assigned to it. A CE includes as its components a Grid Gate (GG) which acts as a generic interface to the cluster; a Local Resource Management System, LRMS (also called batch system) and the cluster itself as a collection of Worker Nodes (WNs), where the jobs actually run.

There are two types of Grid Gate (LCG CE, and gLite CE). From an user viewpoint the CE is the Grid element where jobs gets queued for execution. Formally the "complete name" (CEId) of a CE follows this syntax:

CEId = <gg_hostname>:<port>/<gg_type>-<LRMS_type>-<batch_queue_name>

Thus, recalling the examples that we have seen: prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid is a CE and glite-ce-01.cnaf.infn.it:2119/blah-pbs-egee is another.

4.2 STORAGE ELEMENT (SE)

A Storage Element permits data access in a uniform way, acting as a fronted to the actual storage architecture (disks server, RAID pool, tape based Storage systems and so on). This is achieved thanks to a SRM (Storage Resource Manager) component which provides services for space reservations, files migration (from tape to disk or more complex) and so on. The actual capabilities depend on the specific SRM type, implementation and version in a specific SE. The actual file transfer is performed through a Grid specific protocol (GSIFTP) and common network protocols such as RFIO, in the case of local transfers. Having a SE is not mandatory for a site, nevertheless most sites have one. First Grid's implementation of SE doesn't offer SRM services. See below for a table with types of existing SE and main characteristics.

Type	Resources	File transfer	File I/O	SRM
Classic SE	Disk server	GSIFTP	insecure RFIO	No
DPM	Disk pool	GSIFTP	secure RFIO	Yes
dCache	Disk pool/MSS	GSIFTP	gsidcap	Yes
CASTOR	MSS	GSIFTP	insecure RFIO	Yes

4.3 INFORMATION SERVICE

This is a service intended to provide information about Grid's resources and their status, accessible through LDAP queries and conforming to the GLUE schema. This info is of vital importance for Resource Broker (see below) and for accounting/monitoring purposes. Having hierarchical structure, this info gets collected and published at different levels:

- In a site, at Grid element level (CE, SE, RB), from GRIS services (Grid Resource Information Server). Any GRIS in the site publishes its collected information via LDAP on the 2135 tcp port, intended for local access only.

- at site level, but Grid wide available, from a site-BDII (also known as GIIS: Site Grid Index Information Server) which publishes on the 2170 tcp LDAP port the information collected from the various GRIS. BDII stands for "Berkeley Database Information Index" because of the engine internally used by this Grid element.
- at Grid level, from the "top BDII" which retrieves data from site-BDII, thus providing a complete Grid level knowledge of the Grid status.

4.3.1 Info at a glance

Directly querying a top or site BDII through LDAP queries can give us detailed information (both of static and dynamic nature) about what's exactly available in term of resources. This information can be useful to write down requirements in a JDL for jobs we intend to submit there.

It is possible, indeed, to get information about resources status through web interfaces which permit to get "panoramic view" of Grid status and navigate to refine detail level focusing to specific elements that we are interested in.

One example of these web monitoring tools is GridICE.

See http://gridice2.cnaf.infn.it:50080/gridice/ for an European Grid status view. Specific for the Italian Grid monitoring only there is http://gridice4.cnaf.infn.it:50080/gridice/.

Grid users interested about cyclops VO can select the "VO view" tab and then the "cyclops" link or (time saving shortcut!) directly point to the URL:

http://gridice2.cnaf.infn.it:50080/gridice/vo/vo_details.php?voName=cyclops.

Browsing from there on we can get information about resources related to the cyclops VO (which CE / queues are available at which site, available storage, known jobs in a given time period etc.).

4.4 DATA MANAGEMENT

In a Grid context a file is a sort of "immutable object" and, once created, can only be read or deleted. The reason for this is that a file can have one or more replicas in different Grid sites, and all those copies must remain consistent. This is not a problem for an user viewpoint, since he may refer to files through its Logical File Name (LFN) which Data Management services gets as argument to perform requested operations.

Tipically in the Grid the same file may have one or more different LFNs, a single GUID (Grid Unique Identifier), many SURL (Storage URL) and at least as many TURL (Transport URL). Users may be interested to get a "faster to access" file instead of a replica located in a remote or "narrow band" one. This is why we make mention of all these kind of names used for referring to a same object. Syntaxes are expressed like this:

LFN lfn:<user defined string>, (no mention on physical location)

GUID guid:< 36_chars_unique_string>, (one only per file)

SURL sfn:<SE hostname>/<path> (Classic SE) or srm:<SE hostname>/<path> (SRM based SE)

TURL <protocol>://<SE hostname>:<port>/<path>.

The latter is used (typically by lower level commands) to actually create/read the file, thus it specifies how (protocol, port) and where (SE hostname, path). The mapping between all these names (LFN,GUID,SURL) is kept in a service LFC (LCG File Catalogue) while the files are in the Storage Elements. Note that getting a TURL from a given SURL is performed by asking the srm service in the related SE.

4.5 WORKLOAD MANAGEMENT

The Workload Management System (WMS), who runs in a Resource Broker (RB) is in charge to accept user submitted jobs, assign them to a proper Computing Element, to keep track of their status and retrieve their output.

As we already know, Jobs are described by the Job Description Language (JDL), which can specify, for example, which executable to run and its parameters, files to be moved to and from the Worker Node on which the job is run, input Grid files needed, and requirements on the CE and the Worker Node.

Normally only a subset of the available CEs can effectively handle a certain job. The task of identifying the CE to which send the job is performed by a process called *match-making*. This program firstly determines those CE fulfilling all of the requirements specified in the JDL file, and then it selects the one "closer" to the input Grid files and with the

higher rank, a "goodness" parameter usually based on the load of the CE (which is a function of the number of running and queued jobs).

5 MORE ON JDL

Let's see a few more things about the JDL file needed to define job's requirements. We sketch here a brief explanation about most commonly used attributes, focusing on **submission via WMS WMProxy**. For a complete and more in depth guide on this topic see the EGEE gLite official documentation page: http://glite.web.cern.ch/glite/documentation/ and refer to the **JDL Attributes Specification** guide.

5.1 INTRODUCTION

The Job Description Language (JDL) permits to specify how the Grid must handle stand alone jobs as like as aggregates or collections of jobs with dependency relations.

You can think a JDL file like a set of attributes each one specifying requests, constraints or characteristics. The WMS takes them into account when selecting best resources to run the job. Note that new attributes can be defined from the user itself. A JDL parser will thus validate statements in a syntactic viewpoint only. Attributes which are meaningful to Grid element (specifically to the WMS) are called "supported attributes".

5.2 JDL SYNTAX

A Job Description is a file which components are lines of the form:

attribute = expression;

An expression can span many lines (put a backslash as last char to tell the parser that the expression continues on the next line) and terminates with a semicolon with neither spaces nor tabs after it. A literal string must be enclosed in double quotes. The backslash character can be used to escape double quotes into a literal string. Lines beginning with a sharp (#) or double slash (//) are treated as comments. You can comment a region of text including it between /* ... */ .

5.3 JDL EXAMPLE

We write here an example JDL file with commonly used attributes to let you see how they work. Please be warned that this example does not aim at completeness. Useful attributes may not be mentioned. Refer to more specific documents to get further insight. Explanation of attributes is part of the JDL itself as comments. Please note the Requirements attribute, which expression refers to GLUE schema attributes. Refer to the gLite3 User Guide (appendix G) for a more detailed description (see http://glite.web.cern.ch/glite/documentation/).

```
Executable = "test.sh";
/*
The program to run. It may be an executable already present into the WN, in which case you should need to specify its full path or define custom environment variables.
In this example the program is provided by the user through the InputSandbox
*/
Arguments = "fileA fileB 3";
/*The executable expects two files and a number as arguments. Files mus be provided*/
StdOutput = "std.out";
//This file will catch any standard output message given by the executable
StdError = "std.err";
//This file will catch any standard error message given by the executable
InputSandbox = {"test.sh", "fileA", "fileB"};
/*
```

InputSandbox specifies what files/directories you provide for your job to execute.

You can use jolly characters like this: "my_inp_dir/.dat" in which case any matching*

file present into the directory would be part of the InputSandbox

*/

OutputSandbox = {"std.out", "std.err", "myresults/*"};

/*

OutputSandbox specifies what files/directories you expect to get as execution results from your program. You can use jolly characters like this: "myresults/" in which case any matching file present into the directory would be part of the OutputSandBox*

*/

VirtualOrganisation = "cyclops";

*/*Used to explicitly specify which VO this JDL refers to. Note that this info is implicitly provided by your voms proxy, which also overrides this one*/*

Requirements = other.GlueCEInfoLRMSType == "LFS" && other.GlueCEInfoTotalCPUs > 1;

/* Use the Requirements attribute to specify constraints. You can specify many of them using logical operators such as && (logical AND) || (logical OR) ! (logical NOT). In this case we want LSF as LRMS and the CE which will accept the job must have more than a CPU. This requisite is expressed conforming to the ldap GLUEschema syntax. Here GlueCEInfoLRMSType is the attribute we explicitly refer to and "other" is the prefix who tells the parser to refer to the Information Service. We show below more Requirements examples. Unused ones should be commented out or deleted */

Requirements = other.GlueCEUniqueID == "prod-ce-02.pd.infn.it:2119/blah-lsf-grid";

/*We want our job to be submitted on that particular CE/queue*/

CPU_NEED = other.GlueCEPolicyMaxCPUTime > 120 && other.GlueCEPolicyMaxWallClockTime > 360;

Requirements = CPU_NEED

/*Here we have defined a custom attribute CPU_NEED assigning an expression to it and we have used that attribute to define a requirement: our job may need more than 2 hours CPU time and more than 3 hours Wallclock time. */

Requirements = other.GlueCEPolicyMaxCPUTime > (720 * 1000 / other.GlueHostBenchmarkSI00);

/*This is a finer expression which normalize the time requirement respect to the CPU speed */

Note: if more than one Requirements statement is present, the last one only has effect. Thus you must use logical connectors in order to satisfy many requirements at once.

We have seen how we can transfer small data files needed to run a job or to store its output by defining Input/Output Sandboxes in the JDL file. Large data files, however, should be read and written from/to SEs instead, and registered in a File Catalogue (which maps between LFN, GUID and SURL names), and eventually replicated to more than one SE.

These tasks may be performed using the LCG Data Management client tools. The user should avoid to directly interact with the File Catalogue; he should instead use the LCG tools for File Transfer Service (FTS) which permits to move files between SEs and query the information system to retrieve static or dynamic information about the status of resources and services.

6 DATA MANAGEMENT

Providing needed data files for a job only through the Sandboxes could soon become impractical. For example data files may need too much storage space and/or slow down things. This is why the Grid provides Storage Elements which acts as available storage space for user's data files.

We sketch here a few example steps to get acquainted with data management using SE. The commands involved are similar in concept and behaviour to those of UNIX: ls,mkdir,cp,rm and so on, but since they operate on a completely different underlying infrastructure (the Grid, not a local filesystem!) and they are Grid specific commands instead of OS dependent file manipulation commands, their name is specific: lfc-* commands or lcg-* commands.

6.1 ENVIRONMENT

As usual we first log into a UI, get a proxy certificate and eventually check it:

[sdp@glite-ui]$ voms-proxy-init --voms cyclops

[sdp@glite-ui]$ voms-proxy-info -all

We then get sure that needed enviroment variables are set:

[sdp@glite-ui]$ echo $LCG_GFAL_INFOSYS; echo $LCG_CATALOG_TYPE; echo $LFC_HOST

LCG_GFAL_INFOSYS should already be set. Its value says which top-bdii we are referring to and has the form <top-bdii_fqdn>:port (example: egee-bdii.cnaf.infn.it:2170). Other variables can be set as shown above:

[sdp@glite-ui]$ export LCG_CATALOG_TYPE=lfc

[sdp@glite-ui]$ export LFC_HOST=`lcg-infosites --vo cyclops lfc`

You can put these commands into your ~/.bash_profile file if you want the environment to be set any time you log into the UI. Another comfortable setting would be:

export LFC_HOME=/grid/cyclops/<myworkdir>

which permits us to avoid specifying full path every time.

6.2 THE VIRTUAL FILESPACE

Our files will be visible into a "virtual filesystem" under the "virtual path" /grid/<voname>/. Usually one creates a directory <myusername> and starts putting there its stuff. pay attention that we upload a file into an available SE, thus we need to specify it.

Lets see existing files into the virtual filespace:

[sdp@glite-ui]$ lfc-ls -l /grid/cyclops

drwxrwxr-x 3 237 130 0 Apr 17 16:24 adirname

Create a directory for datafiles:

[sdp@glite-ui]$ lfc-mkdir /grid/cyclops/$USER

We can see the dir repeating the lfc-ls command. Also we can see which right we have on this directory. This is the analogous of the UNIX ls -la command, but quite more informative.

[sdp@glite-ui]$ lfc-getacl /grid/cyclops/sdalpra/

file: /grid/cyclops/sdalpra/

owner: /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Stefano Dal Pra

group: cyclops

user::rwx

group::rwx #effective:rwx

other::r-x

default:user::rwx

default:group::rwx

default:other::r-x

6.3 TRANSFER FILES TO THE GRID

We want now to transfer some file into the Grid, namely into a Storage Element. First we look for available SEs:

[sdp@glite-ui]$ lcg-infosites --vo cyclops se

Avail Space(Kb) Used Space(Kb) Type SEs

395252028	4874600	n.a	gridse.ilc.cnr.it
479999416	1329338248	n.a	grid007g.cnaf.infn.it
3937229840	456948720	n.a	prod-se-01.pd.infn.it
52670524	656381100	n.a	gridit002.pd.infn.it
3850000000	240000000	n.a	prod-se-02.pd.infn.it
67016288	4009492	n.a	gridse.sns.it

6.3.1 lcg-cr (copy & register)

We can transfer a local file into an SE, say prod-se-02.pd.infn.it. see the "copy and register" command below:

[sdp@glite-ui]$ lcg-cr --vo cyclops -d prod-se-02.pd.infn.it \

file:///home_local/sdalpra/cpi -l lfn:///grid/cyclops/$USER/cpi_test_executable

guid:33885195-3b15-497f-bd88-f193a9243704

We specified the desired Storage Element, the local file to transfer (using the file:// protocol) and the desired "logical file name" (using the lfn:// protocol). After the command succeeds it returns the GUID of the file.

6.3.2 lcg-rep (create a replica)

We want to put a copy of the same file into another available SE too. We thus create a replica of the file:

[sdp@glite-ui]$ lcg-rep --vo cyclops -d gridit002.pd.infn.it \
lfn:///grid/cyclops/$USER/cpi_test_executable

6.3.3 links

Symbolic links can be created also; they are very similar to links in UNIX:

lfc-ln -s /grid/cyclops/sdalpra/cpi_test_executable /grid/cyclops/sdalpra/a_link

lfc-ln -s guid:33885195-3b15-497f-bd88-f193a9243704 /grid/cyclops/sdalpra/another_link

6.4 GET FILE INFORMATION

Commands to get file informations are based on the File Catalog service

6.4.1 lcg-lr (list replicas)

Have a look of existing replicas:

[sdp@glite-ui]$ lcg-lr --vo cyclops lfn:///grid/cyclops/sdalpra/cpi_test_executable
srm://prod-se-02.pd.infn.it/dpm/pd.infn.it/home/cyclops/generated/2007-04-24/filea51a6418-ef78-485d-8df1-4747337049ff
sfn://gridit002.pd.infn.it/flatfiles/SE00/cyclops/generated/2007-04-24/filef8d5b2b7-2660-496e-8b5a-9f5b94404dca

You see the two SURL of the replicas of our file, one per SE. Note that lcg-lr accepts also the GUID as argument:

[sdp@glite-ui]$ lcg-lr --vo cyclops guid:33885195-3b15-497f-bd88-f193a9243704

6.4.2 lcg-lg (list the GUID)

Also, you can get the file GUID given a lfn or a SURL:

[sdp@glite-ui]$ lcg-lg --vo cyclops lfn:///grid/cyclops/sdalpra/rocrep_tgz.tgz

guid:da2704f4-3176-43ab-9374-cbd87095464a

[sdp@glite-ui]$ lcg-lg --vo cyclops \

srm://prod-se-02.pd.infn.it/dpm/pd.infn.it/home/cyclops/generated/2007-04-26/filef66675c4-1e55-40bf-87de-3c9bb75faf13

guid:da2704f4-3176-43ab-9374-cbd87095464a

6.4.3 lcg-gt (get the TURL)

The commands shown above make all use of the File Catalog service in order to perform the mappings. To get a TURL, however, an actual data transfer protocol (gsiftp, rfio, dcap) need to be also specified. See the command below:

```
[sdp@glite-ui]$ lcg-gt \
srm://prod-se-02.pd.infn.it/dpm/pd.infn.it/home/cyclops/generated/2007-04-26/filef66675c4-1e55-40bf-87de-3c9bb75faf13 rfio
rfio://prod-se-02.pd.infn.it//flatfiles/SE00/cyclops/2007-04-26/filef66675c4-1e55-40bf-87de-3c9bb75faf13.2641133.0
2860099
0
```

The last two lines give the information needed to set the file state to “Running” or “Done” in the Storage Resource Manager (they are only meaningful if the space is managed by an SRM).

```
[sdp@glite-ui]$ lcg-gt \
srm://prod-se-02.pd.infn.it/dpm/pd.infn.it/home/cyclops/generated/2007-04-26/filef66675c4-1e55-40bf-87de-3c9bb75faf13 gsiftp
gsiftp://prod-se-02.pd.infn.it/prod-se-02.pd.infn.it:/flatfiles/SE00/cyclops/2007-04-26/filef66675c4-1e55-40bf-87de-3c9bb75faf13.2641133.0
2860202
0
```

6.5 TRANSFER FILES FROM THE GRID

Get back the file. There are plenty of ways to retrieve a file from the Grid:

1. specifying a lfn:

   ```
   lcg-cp --vo cyclops lfn:///grid/cyclops/sdalpra/rocrep_tgz.tgz file://$PWD/getbacklfn
   ```

2. specifying the GUID

   ```
   lcg-cp --vo cyclops guid:33885195-3b15-497f-bd88-f193a9243704 file://$PWD/getbackfile
   ```

3. specifying a SURL:

   ```
   lcg-cp --vo cyclops \
   srm://prod-se-02.pd.infn.it/dpm/pd.infn.it/home/cyclops/generated/2007-04-24/filea51a6418-ef78-485d-8df1-4747337049ff
   file://$PWD/getbacksrm
   ```

4. Specifying a TURL:

   ```
   lcg-cp --vo cyclops \
   gsiftp://prod-se-02.pd.infn.it/prod-se-02.pd.infn.it:/flatfiles/SE00/cyclops/2007-04-26/filef66675c4-1e55-40bf-87de-
   3c9bb75faf13.2641133.0 file://$PWD/getbacklfn2
   ```

Note how only the last two ways specifies a physical copy of the file. First one does not care about from where the file actually gets retrieved. The third method is thus faster since it skips the “decision making” step needed to choose the best source to gets the file from. Real applications however relies on “closest SE” criterion, implemented in order to automatically find the best SE to transfer the file from.

6.6 GRID FILES AND JDL

Now we would like to specify in a jdl file how to handle our Grid datafile. We illustrate a piece of jdl with comments to illustrate the meaning of the attributes.

```
Executable = "script.sh";
// The script to execute
Arguments = "argumentstring";
```

```
// The argument to pass to it (referred as $1 into the script itself)
 InputSandbox = {"script.sh"};
 InputData = {"lfn:///grid/cyclops/<myworkdir>/inputfilename"};
//the input file, previously uploaded in the Grid.
//The executable script will need first of all to copy
//this file into the WN when executing, thus issuing:
//lcg-cp --vo cyclops lfn:///grid/cyclops/<myworkdir>/inputfilename file://$PWD/inputfilename
 DataAccessProtocol = {"gridftp","rfio","gsiftp"};
//the TURL to access the file may use one of these protocols
 OutputSE = "prod-se-02.pd.infn.it";
//Where to store script output
 DataCatalog = "http://lfcserver.cnaf.infn.it:8085/";
//Optional. The file catalog to use. Default one should be good.
 OutputData = {
                [
                    OutputFile = "upload.out";
                    LogicalFileName = "lfn:/grid/cyclops/upload.out";
                    StorageElement = "prod-se-02.pd.infn.it";
                ]
        }
//Executable will produce upload.out as output file. We assign a lfn to it and a SE.
//The 'copy and register" of the output file will be executed with the given information:
//lcg-cr --vo cyclops \
//file://$PWD/upload.out -d prod-se-02.pd.infn.it -l lfn:/grid/cyclops/<myworkdir>/upload.out
//If StorageElement is not specified the SE will be automatically selected
//by the WMS, as a close one to the CE.
OutputSandbox = {"std.out","std.err"};
//The Output sandbox will contain a further file with the logs of the output upload.
//There you'll read the file GUID in case of success.
```

7 SPECIAL JOB TYPES

7.1 MPI JOBS

The "Message Passing Interface" (see http://www-unix.mcs.anl.gov/mpi/ for details) permits to write software (Executable, in jdl terms) that runs in parallel on several CPU. In a JDL you specify this through the **JobType** ("MPICH") and the **NodeNumber** attribute (who tells how many CPU-cores are required to run the job). For a more detailed explanation see http://grid-it.cnaf.infn.it/index.php?mpihowto&type=1.

Here we provide and test the example jdl file below which provides as output an estimation of pi performed by a parallel execution of the cpi executable.

Files needed to test following example can be downloaded from:

http://forge.cnaf.infn.it/plugins/scmsvn/viewcvs.php/trunk/ig-certification/mpi/new/?root=igrelease

```
Type = "job";

JobType = "mpich";

NodeNumber = 4;

Executable = "cpi";

StdOutput = "sim.out";

StdError = "sim.err";

OutputSandbox = {"sim.err","sim.out"};
```

```
RetryCount = 3;
InputSandbox = {"cpi"};
rank = other.GlueCEStateFreeCPUs;
```

As usual we check for available CEs using glite-wms-job-list-match:

```
[sdp@glite-ui]$ glite-wms-job-list-match –a MPItest.jdl
Connecting to the service https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server
================================================================================
                COMPUTING ELEMENT IDs LIST
 The following CE(s) matching your job requirements have been found:
       *CEId*
 - grid003.roma2.infn.it:2119/jobmanager-lcgpbs-grid
 - gridce.sns.it:2119/jobmanager-lcgpbs-grid
 - prod-ce-01.pd.infn.it:2119/jobmanager-lcglsf-grid
================================================================================
```

and we the submit it:

```
[sdp@glite-ui]$ glite-wms-job-submit -a -o mpi_jobid.txt MPItest.jdl
Connecting to the service https://prod-wms-01.pd.infn.it:7443/glite_wms_wmproxy_server
====================== glite-wms-job-submit Success ======================
The job has been successfully submitted to the WMProxy
Your job identifier is:
https://prod-wms-01.pd.infn.it:9000/tm1TCpG-KV-hLjcEPs_S4Q
The job identifier has been saved in the following file:
/home/sdalpra/mpi_jobid.txt
================================================================================
```

After the job finishes we retrieve the output:

```
[sdp@glite-ui]$ glite-wms-job-output --dir mpi_output \
 https://prod-wms-01.pd.infn.it:9000/tm1TCpG-KV-hLjcEPs_S4Q
Connecting to the service https://193.206.210.111:7443/glite_wms_wmproxy_server
================================================================================
                JOB GET OUTPUT OUTCOME
Output sandbox files for the job:
https://prod-wms-01.pd.infn.it:9000/tm1TCpG-KV-hLjcEPs_S4Q
have been successfully retrieved and stored in the directory:
/home/sdalpra/mpi_output
```

Finally we look at the mpi_output dir, where we see the executable.out which contains an estimation of pi:

```
[sdp@glite-ui mpi_output]$ cat executable.out
Process 0 of 4 on prod-wn-03.pd.infn.it
pi is approximately 3.1415926544231239, Error is 0.0000000008333307
```

```
wall clock time = 10.008569
Process 2 of 4 on prod-wn-03.pd.infn.it
Process 1 of 4 on prod-wn-04.pd.infn.it
Process 3 of 4 on prod-wn-05.pd.infn.it
```

7.2 JOB COLLECTIONS

It is possible to submit as a single unit a bunch of distinct jobs and see them all as a single entity. This may be useful when all sub-jobs refer to a same input dataset. WMS allows sharing of sandboxes and inheritance, thus permitting useful optimizations, such as transferring a file needed to many sub-jobs once only.

In the following jdl example an InputSandbox is defined and three jobs (*node1*, *node2*, *node3*) are specified under the "node" section. They in turn define their own InputSandbox, making use of part or all of the main one (root.InputSandbox[0], is the first file of the main input sandbox, root.InputSandbox is the whole) so that node1 and node2 inherits one file each, and node3 inherits the full InputSandbox.

```
[
  Type = "collection";
  InputSandbox = {
    "input_common1.txt",
    "input_common2.txt"
  };
nodes = {
   [
     JobType = "Normal";
     NodeName = "node1";
     Executable = "/bin/sh";
     Arguments = "script_node1.sh";
     InputSandbox = {"script_node1.sh",
                     root.InputSandbox[0]
                    };
     StdOutput = "myoutput1";
     StdError  = "myerror1";
     OutputSandbox = {"myoutput1","myerror1"};
   ],[
     JobType = "Normal";
     NodeName = "node2";
     Executable = "/bin/sh";
     InputSandbox = {"script_node2.sh",
                     root.InputSandbox[1]
                    };
     Arguments = "script_node2.sh";
```

```
    StdOutput = "myoutput2";
    StdError  = "myerror2";
    OutputSandbox = {"myoutput2","myerror2"};
  ],[
    JobType = "Normal";
    NodeName = "node3";
    Executable = "/bin/cat";
    InputSandbox = {root.InputSandbox};
    Arguments = "*.txt";
    StdOutput = "myoutput3";
    StdError  = "myerror3";
    OutputSandbox = {"myoutput3","myerror3"};
  ]
 };
]
```

7.3 DAG JOBS

One may want to set up a set of rules to define dependencies between jobs, i.e. having as input for a job the output of one or more other jobs; in order to achieve this, an order of execution for jobs must be defined. In the example given below, four nodes are defined: "*father*, *son1*, *son2*, *final*".

A common InputSandbox is defined and individual sub-jobs refer to it with the same notation seen for the Job collections example.

The dependencies attribute defines the order of execution: the {A,B} notation tells that A must have finished before B starts. Note how the requirements can be nested.

Here is the jdl file:

```
[
  Type = "dag";
  InputSandbox = {"son.sh"};
nodes = [
   father = [
    description = [
     JobType = "Normal";
     Executable = "/bin/sh";
     Arguments = "father_script.sh";
     InputSandbox = {"father_script.sh"};
     StdOutput = "father_output";
     StdError  = "father_error";
     OutputSandbox = {"father_output","father_error","son1.input","son2.input"};
    ];
  ];
  son1 =  [
   description = [
     JobType = "Normal";
     Executable = "/bin/sh";
     InputSandbox = {root.InputSandbox,root.nodes.father.description.OutputSandbox[2]};
     Arguments = "son.sh 1";
     StdOutput = "son1.output";
```

```
    StdError  = "son1.error";
    OutputSandbox = {"final1.input","son1.output","son1.error"};
   ];
  ];
   son2  =  [
    description = [
     JobType = "Normal";
     Executable = "/bin/sh";
     InputSandbox = {root.InputSandbox,root.nodes.father.description.OutputSandbox[3]};
     Arguments = "son.sh 2";
     StdOutput = "son2.output";
     StdError  = "son2.error";
     OutputSandbox = {"final2.input","son2.output","son2.error"};
    ];
   ];
   final =  [
    description = [
     JobType = "Normal";
     Executable = "/bin/sh";
     InputSandbox = {"final.sh",root.nodes.son1.description.OutputSandbox[0],
                root.nodes.son2.description.OutputSandbox[0]};
     Arguments = "final.sh";
     StdOutput = "dag.out";
     StdError  = "dag.err";
     OutputSandbox = {"dag.out","dag.err"};
    ];
   ];
    dependencies =  {
       {father,{son1,son2}},
       {son1,final}, {son2,final}
     };
 ];
]
```

8 CONCLUSION

After reading this document a cyclops VO user should be able to write and test simple jobs, and should have a “first order” understanding about various Grid components involved. To reach a more detailed comprehension further readings are strongly recommended.

A good practical source of explained examples can be found on the GILDA wiki site:

https://grid.ct.infn.it/twiki/bin/view/GILDA/WebHome

Also, the gLite User Guide (version 3.0 at the time of this writing, see:

https://edms.cern.ch/file/722398/1.1/gLite-3-UserGuide.html)

is an introduction to the WLCG/EGEE Grid and to the gLite 3 middleware from a user’s point of view.

The official repository of gLite Grid documentation (See the “User Manuals” section) is here:

http://glite.web.cern.ch/glite/documentation/.

Last but not least, the ggus portal, with FAQ/Wiki and Documentation section.

http://www.ggus.org