EGI-InSPIRE

HYDRA SERVICE DEPLOYMENT ON A MULTI-SERVERS CONFIGURATION

EU MILESTONE: MS607

Document identifier:	EGI-MS607-327-v0.2.doc
Date:	**10/06/2011**
Activity:	SA3
Lead Partner:	CNRS
Document Status:	FINAL
Dissemination Level:	PUBLIC
Document Link:	https://documents.egi.eu/document/327

Abstract

The document provides details about the deployment of the Hydra file encryption/decryption service in EGI and provides update on the status of this activity. Hydra will be deployed as a distributed server-triplet to store segments of security users' keys for encrypting and decrypting files located on grid storage services. Using Shamir's secret sharing algorithm Hydra will provide a secure and fault-tolerant environment file encryption service for communities that need highly secured file access on EGI. A Hydra installation for the Life Sciences Heavy User Community of EGI will be also provided by the same team. Progress with the activities is currently pending on waiting for the Hydra software to be ready for gLite 3.2 by the EMI project.

I. COPYRIGHT NOTICE

II. DELIVERY SLIP

	Name	Partner/Activity	Date
From	Franck Michel	CNRS / SA3	22-04-2011
Reviewed by	Moderator: Gergely Sipos Reviewers: Christos Kanellopoulos	EGI.eu/NA3 GRNET	09-06-2011
Approved by	AMB & PMB		10-06-2011

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	22-04-2011	First draft	Franck Michel
2	09-06-2011	Updated after review	Franck Michel
3	10-06-2011	Final version	Franck Michel

IV. APPLICATION AREA

This document is a formal deliverable for the European Commission, applicable to all members of the EGI-InSPIRE project, beneficiaries and Joint Research Unit members, as well as its collaborating projects.

V. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the authors. The procedures documented in the EGI-InSPIRE "Document Management Procedure" will be followed: https://wiki.egi.eu/wiki/Procedures

VI. TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/.

VII. PROJECT SUMMARY

To support science and innovation, a lasting operational model for e-Science is needed – both for coordinating the infrastructure and for delivering integrated services that cross national borders.

The EGI-InSPIRE project will support the transition from a project-based system to a sustainable pan-European e-Infrastructure, by supporting 'grids' of high-performance computing (HPC) and high-throughput computing (HTC) resources. EGI-InSPIRE will also be ideally placed to integrate new Distributed Computing Infrastructures (DCIs) such as clouds, supercomputing networks and desktop grids, to benefit user communities within the European Research Area.

EGI-InSPIRE will collect user requirements and provide support for the current and potential new user communities, for example within the ESFRI projects. Additional support will also be given to the current heavy users of the infrastructure, such as high energy physics, computational chemistry and life sciences, as they move their critical services and tools from a centralised support model to one driven by their own individual communities.

The objectives of the project are:

1. The continued operation and expansion of today's production infrastructure by transitioning to a governance model and operational infrastructure that can be increasingly sustained outside of specific project funding.
2. The continued support of researchers within Europe and their international collaborators that are using the current production infrastructure.
3. The support for current heavy users of the infrastructure in earth science, astronomy and astrophysics, fusion, computational chemistry and materials science technology, life sciences and high energy physics as they move to sustainable support models for their own communities.
4. Interfaces that expand access to new user communities including new potential heavy users of the infrastructure from the ESFRI projects.
5. Mechanisms to integrate existing infrastructure providers in Europe and around the world into the production infrastructure, so as to provide transparent access to all authorised users.
6. Establish processes and procedures to allow the integration of new DCI technologies (e.g. clouds, volunteer desktop grids) and heterogeneous resources (e.g. HTC and HPC) into a seamless production infrastructure as they mature and demonstrate value to the EGI community.

The EGI community is a federation of independent national and community resource providers, whose resources support specific research communities and international collaborators both within Europe and worldwide. EGI.eu, coordinator of EGI-InSPIRE, brings together partner institutions established within the community to provide a set of essential human and technical services that enable secure integrated access to distributed resources on behalf of the community.

The production infrastructure supports Virtual Research Communities (VRCs) – structured international user communities – that are grouped into specific research domains. VRCs are formally represented within EGI at both a technical and strategic level.

VIII. EXECUTIVE SUMMARY

As part of the services delivered to the Life Sciences HUC, CNRS will deploy a service for sensitive files encryption/decryption based on the Hydra service, a key-based storage solution where encryption keys are stored in a distributed key store, ensuring security and fault-tolerance. The Hydra software is developed at CERN and currently migrated to gLite 3.2.

The deployment consists in the software installation and configuration on three separated key stores and the publication of the service in the BDII, the pre-installation of the client side software on every sites in the scope of the LS HUC VOs as well as on users' desktops, and the set up of a system to monitor the quality of the service. This deliverable provides procedures for those different tasks.

An experimental Hydra service has been deployed on a gLite release 3.1 UI. The migration of Hydra to gLite release 3.2 is currently ongoing within the EMI project. The details of the installation and operation procedure are being worked out with the EMI support team. Once completed, the next actions will include the provision and installation of the three key stores-based encryption service, the pre-installation and publication of the Hydra client on every site, and the publication of a test suite to periodically assess the quality of the service delivered.

TABLE OF CONTENTS

1 INTRODUCTION

The Hydra service [R2] is an encrypted storage solution enabling encryption of files stored on storage resources. The sensitive information - the encryption and decryption keys - is stored in the Hydra key store that ensures a secure and controlled access to them. Hydra splits and distributes key pieces to several (e.g. three) key stores, according to the principle of the Shamir's secret sharing algorithm [R1]. The key pieces are:

1. Partially redundant (e.g. 2 out of 3 pieces are needed to reconstruct an encryption key)
2. Always incomplete (e.g. at least 2 pieces are needed).

Thus even if one server fails or is compromised, the keys are still accessible and protected (see [R1]).

The Hydra service is provided in task TSA3.2.3 within the SA3 "Services for the Heavy User Communities" (HUC) work package of EGI-InSPIRE. This task covers the provisioning of all needed hardware and software for the Life Sciences HUC, along with installation and configuration documentation for any EGI user community. This document describes the status of the ongoing work on these activities..

2 DEPLOYMENT FOR THE LIFE SCIENCES HUC

2.1 Use case

The following use case illustrates the typical usage of the Hydra service from the end-user side point of view.

A user wishes to run a job that processes input files and produces result files. Input and output files may contain sensitive data (e.g. private medical information). The procedure shall perform the following steps:

1. Encrypt files to be processed using the user's key id and proxy certificate: the Hydra service retrieves user's key pieces from two of the three key stores, rebuilds the key and encrypts the files.
2. Register the encrypted files on some Storage Element.
3. Submit the job that shall:
    3.1 Get the encrypted files, use the Hydra service to decrypt them locally on the Worker Node using the user's key id like in step 1,
    3.2 Perform the application specific processing of decrypted files, then encrypt produced files so that they can be securely transferred back to the user thought the job sandbox.
4. The user retrieves the encrypted result files from the Workload Management Service via the User Interface, and uses the Hydra service to decrypt them locally.

2.2 Targeted deployment

As outlined in section 1, the Hydra service for the Life Sciences HUC will be deployed on three separate servers (key stores) where keys are distributed, in order to provide both security (if a server is compromised, it does not compromise the keys privacy), and fault tolerance (two servers are sufficient to reconstruct a key). Table 1 describes the three servers:

Server hostname	Release	Hosting site
egee2.unice.fr	gLite UI 3.2 / CentOS5.5 64 bits	I3S, CNRS, France
vip.creatis.insa-lyon.fr	gLite UI 3.2 / CentOS5.5 64 bits	Creatis, CNRS, France
ui.lsgc.healthgrid.org	gLite UI 3.2 / SL5.5 64 bits	HealthGrid, France

Table 1: Hydra key stores for the LS HUC

The Hydra service must be published in the BDII using a unique ID. From the user's point of view, the service may be configured either through this unique ID published in the BDII, or using a local services.xml file. The first is the preferred and most simple solution for users, however the latter remains possible and both solutions are described in this document.

2.3 Choice of the target platform

The Hydra software packages are available for gLite 3.0 and 3.1, however these are not satisfactory for the Life Sciences community, because the community members already migrated to gLite 3.2, a secure keystore should be provided for them on this platform. Work is currently ongoing to migrate Hydra to gLite 3.2 by the gLite experts from the EMI project working for CERN, who are responsible for Hydra development. No availability date has been provided yet, although recent communications

with EMI indicate that the software will be ready soon (around the time this milestone document will be finalised).

Current gLite 3.1 UIs are often installed on 32-bits machines. Sooner or later laboratories will need to upgrade to gLite 3.2 which also means that machines will need to be changed to 64 bits platforms. Therefore, for long term maintenance issues, it is deemed preferable to install the Hydra service on 64-bits gLite 3.2 UIs right now, rather than have to move installed key stores on new machines later on. Additionally the three sites (previous section) that will host Hydra servers have gLite 3.2 UIs available for this purpose.

2.4 Tasks and status

Table 2 describes the different tasks to be performed to deploy the Hydra service, along with their current status.

#	Task description	Responsible	Current status
1	Hydra software provisioning + installation and configuration documentation	EMI (CERN)	Operational for gLite 3.0 and 3.1. Official installation and configuration procedure available [R3] for gLite 3.0. Unofficial customized procedure [R4] applies to gLite 3.1/SL4. The migration of Hydra software packages to gLite 3.2 is ongoing. No delivery date provided at this time, no installation and deployment procedure available, despite CNRS requests for date and status. Status: EMI (CERN) development team to provide availability date asap.
2	Hardware provisioning	CNRS, Health Grid	Three gLite UIs are installed and ready for deployment on three separate sites in France (see section 2.1). Status: completed.
3	Provide a customized installation and configuration procedure for administrators to deploy the Hydra key store for the LS HUC.	CNRS	This procedure will derive from the installation and configuration procedure provided by the EMI team that develops Hydra (task 1). Status: waiting for procedure from EMI team.
4	Install and configure Hydra and third party software packages on each Hydra server: Apache Tomcat, gLite Trust Manager, MySQL, Hydra server packages.	CNRS	Status: an experimental Hydra service has been deployed on server egee1.unice.fr (gLite 3.1/SL4). Users are configured using the local file configuration mode (see section 6.1), rather than the Service Discovery mode. A procedure for gLite 3.2 is currently being worked out (see section 5.1) with the support from the EMI team involved in this migration. As soon as this is completed, the operational key store service will be deployed on three separate servers. Waiting for availability date from EMI (CERN) team.

#	Task description	Responsible	Current status
5	Make Hydra client package available to all Worker Nodes by pre-installing it on the shared area of each site accessible to the LS HUC VOs, and publishing the corresponding runtime environment tag in the BDII for those sites.	CNRS	Status: procedure available (see section 5.2), to be tested and deployed.
6	Publish the Hydra service in the BDII so that users can access it through the gLite Service Discovery [R6] simply based on the service unique id.	CNRS	The registration of the Hydra service shall be done only when the production service will be installed (task 4). Until then, the experimental service uses the local-file based configuration. Status: ongoing; procedure to be clarified with BDII administrators (see section 5.3).
7	Provide users with an installation and usage procedure: install the Hydra client package on gLite UIs, Hydra client commands usage manual.	CNRS	Status: usage procedure available (see section 6.3), a more detailed example to be provided. Procedure for the local file configuration available (see section 6.1). Procedure for the BDII configuration to be completed (see section 6.2) when the Hydra service is ready to be launched.
8	Monitoring: set up probes to (1) monitor the service from an end-user point of view (encryption/decryption commands succeed), and (2) make sure each keystore server is up and running.	CNRS	To be studies how the Biomed Nagios box may be used to host dedicated probes. Status: to be done. Not critical for the service launch, can be addressed later on.

Table 2: Deployment tasks and status

3 CONCLUSION

As part of the services delivered to the Life Sciences HUC, CNRS will deploy a service for sensitive files encryption/decryption based on the Hydra service.

An experimental Hydra service has been successfully deployed on a gLite release 3.1 UI and is usable for test purposes. Work is currently on going to migrate Hydra to gLite release 3.2 at CERN by gLite experts from the EMI project. This deliverable describes a procedure to achieve deployment of older Hydra packages and configure them to use on the EGI infrastructure. However this procedure has not proved to be fully operational and is being worked out with the support from the CERN team involved in this migration.

Beside deployment of the key store services, it is needed to install and publish the Hydra client on every site where Worker Nodes may be required to access the Hydra service (in this case, all sites accessible to the LS HUC VOs). Similarly, client desktops using the Hydra service should be configured as described in this deliverable.

The next actions concerning this task include the provisioning and installation of a three-key store-based service on gLite 3.2 servers, the installation and publication of the Hydra client on every sites in the scope of the LS HUC VOs, and the publication of a test suite to periodically assess the quality of the service delivered.

4 REFERENCES

R 1	A. Shamir. "How to share a secret" in Communications of the ACM (CACM), vol. 22, pages 612-613, 1979.
R 2	Hydra service overview. https://twiki.cern.ch/twiki/bin/view/EGEE/DMEDS
R 3	Hydra service installation. http://glite.web.cern.ch/glite/packages/R3.0/R20060502/doc/installation_guide_3.0-2.html#_Toc135537608
R 4	Hydra test set-up installation for gLite 3.1/SL4 http://euindia.ictp.it/stock-analysis-application/hydra-test-set-up-installation
R 5	Software installation, Shared Area Method http://www.eu-egee.org/fileadmin/documents/UseCases/SWInstallation.html#id2512589
R 6	Service Discovery User Guide: https://edms.cern.ch/document/578147/1

5 APPENDIX - HYDRA SERVICE INSTALLATION

The latest official installation and configuration procedure available [R3] was written for gLite 3.0. An unofficial customized procedure [R4] applies to gLite 3.1/SL4, it has been reused and adapted in the context of the Life Sciences HUC to provide the procedure hereafter for gLite 3.2.

Note that this procedure has not proved to be fully operational, it is being worked out with the support from the CERN team involved in this migration.

5.1 Hydra server installation and configuration procedure

This procedure applies to a UI installed with CentOS 5.5 64 bits and gLite 3.2. It consists in several steps described hereafter. To be fully functional, each Hydra key store server requires the following software components:

- Apache Tomcat servlet container to host the web services to access the key store;
- gLite Trust Manager to enable Tomcat-hosted applications to interact with the Grid Security Infrastructure (GSI);
- MySQL database server to store keys and ACLs;
- Hydra server specific packages.

5.1.1 Install Tomcat5

5.1.1.1 Install packages

As root run the command:

```
[root]# yum install tomcat5 tomcat5-webapps tomcat5-admin-webapps
```

Then register tomcat5 to start at machine start-up, and start the service:

```
[root]# chkconfig --level 345 tomcat5 on
[root]# service tomcat5 start
```

Make sure that $CATALINA_HOME and $CATALINA_BASE environment variables are set properly, e.g. with value /usr/share/tomcat5/.

5.1.1.2 Configure the gLite Trustmanager

In order to facilitate the Tomcat configuration the Trustmanager package needs to be updated: version 2.5.5.1 is currently installed along with gLite 3.2, the most recent at the time we make this installation is the 2.5.5.4. This latest version comes with an additional package that configures tomcat5 dependencies.

As root, add a new Yum repository if it does not exist yet: /etc/yum.repos.d/epel.repo with the following content:

```
[epel]
name=EPEL
baseurl=http://download.fedora.redhat.com/pub/epel/5/$basearch
gpgkey=http://download.fedora.redhat.com/pub/epel/RPM-GPG-KEY-EPEL
gpgcheck=1
```

```
enabled=1
```

Then use yum to install the following packages: glite-security-util, glite-security-trustmanager, glite-security-trustmanager-tomcat5.

Configure the gLite Trusmanager to be used as the security layer in Tomcat. It is crucial to configure gLite Trustmanager by running the script /opt/glite/etc/glite-security-trustmanager/configure.sh: this will copy proper jars in Tomcat $CATALINA_HOME/server/lib directory.

5.1.1.3 Server certificate

Make a copy of the host certificate and private key to be readable by user *tomcat*. If the host certificate does not exist yet, it must be requested to the appropriate Certification Authority, namely GRID2-FR in our case. As root, do the following:

```
[root]# cd /etc/grid-security
[root]# cp hostcert.pem tomcatcert.pem
[root]# cp hostkey.pem tomcatkey.pem
[root]# chown tomcat:tomcat tomcat*pem
[root]# chmod 600 tomcatkey.pem
```

5.1.1.4 Configure server.xml

Copy file /opt/glite/etc/glite-security-trustmanager/server.xml provided by the Trustmanager packages into the Tomcat configuration folder: $CATALINA_HOME/conf/server.xml.

Update the provided connector section to set the proper certificate and private key files:

```
<Server port="8005" shutdown="SHUTDOWN">
 <Service name="Catalina">

  <Connector port="8443" SSLEnabled="true"
   maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
   enableLookups="false" disableUploadTimeout="true"
   acceptCount="100" debug="0" scheme="https" secure="true"
   sSLImplementation=
     "org.glite.security.trustmanager.tomcat.TMSSLImplementation"
   trustStoreDir="/etc/grid-security/certificates"
   sslCertFile="/etc/grid-security/tomcatcert.pem"
   sslKey="/etc/grid-security/tomcatkey.pem"
   crlUpdateInterval="2h"
   log4jConfFile="/usr/share/tomcat5/conf/log4j-trustmanager.properties"
   clientAuth="true" sslProtocol="TLS"
   crlEnabled="true" crlRequired="true"/>

  <Engine name="Catalina" defaultHost="localhost">
   <Host name="localhost" appBase="webapps" />
  </Engine>
 </Service>
```

5.1.2 Install MySQL

5.1.2.1 MySQL core packages

First install the basic MySQL packages. As root run the command:

```
[root]# yum groupinstall mysql
```

Then configure the root password:

```
[root]# /usr/bin/mysqladmin -u root password 'choose_a_password'
[root]# /usr/bin/mysqladmin -u root -p -h egee2.unice.fr password 'choose_a_password'
Enter password: ******
```

Register mysqld to start at machine startup, and start the service:

```
[root]# chkconfig --level 2345 mysqld on
[root]# service mysqld start
```

5.1.2.2 Optional GUI Tools

Optionally, install the GUI Tools that in particular comprise the graphical MySQL Administrator.
Note that this should be installed on a test environment only, but not on a production server.

```
[root]# wget http://dev.mysql.com/get/Downloads/MySQLGUITools/mysql-gui-tools-5.0r12-rhel4-i386.tar.gz/from/http://mirrors.ircam.fr/pub/mysql/
(...)
[root]# gunzip mysql-gui-tools-5.0r12-rhel4-i386.tar.gz
[root]# tar xvf mysql-gui-tools-5.0r12-rhel4-i386.tar
[root]# rpm -ivh mysql-administrator-5.0r12-1rhel4.i386.rpm mysql-gui-tools-5.0r12-1rhel4.i386.rpm mysql-query-browser-5.0r12-1rhel4.i386.rpm
(...)
```

5.1.2.3 JDBC connector

Also download and install the MySQL JDBC connector package, and copy that connector jar into Tomcat lib folder:

```
[root]# wget http://mirror.centos.org/centos/5/os/i386/CentOS/libgcj-4.1.2-48.el5.i386.rpm
(...)
[root]# rpm -ivf libgcj-4.1.2-48.el5.i386.rpm
(...)
[root]# rpm -ivh mysql-connector-java-5.1.12-2.el5.i386.rpm
[root]# cp /usr/share/java/mysql-connector-java-5.1.12.jar /usr/share/tomcat5/common/lib/
```

5.1.3 Install Hydra

5.1.3.1 Install packages

Get the last Hydra packages: package org.glite.data.hydra-service for the key store server side, and package org.glite.data.hydra-cli for the glite-eds-* client commands. The following locations have been

used for the prototype deployment, however final locations for production packages should be provided by the developers team:

http://eticssoft.web.cern.ch/eticssoft/repository/org.glite/org.glite.data.hydra-service/

http://eticssoft.web.cern.ch/eticssoft/repository/org.glite/org.glite.data.hydra-cli/

Install them:

```
[root]# rpm -ivh glite-data-hydra-service-1.3.5-1sl5.noarch.rpm
[root]# rpm -ivh glite-data-hydra-cli-3.1.2-4.sl5.x86_64.rpm
```

Copy the Hydra war /opt/glite/share/java/glite-data-hydra-service.war to $CATALINA_HOME/webapps, and check that the webapp has been expanded correctly by checking the following url with any html browser: https://hostname:8443/glite-data-hydra-service/

5.1.3.2 Configuration

In directory /opt/glite/etc/glite-data-hydra-service, copy file config.properties.example to config.properties and customize it as needed.

A very simple file with only one key repository may look like this:

```
HYDRA_INSTANCES="1"
HYDRA_DBNAME_1=hydradb1
HYDRA_DBUSER_1=hydrausr1
HYDRA_DBPASSWORD_1=hydra1
HYDRA_CREATE_1=/biomed/Role=NULL/Capability=NULL
HYDRA_ADMIN_1="/O=GRID-FR/C=FR/O=CNRS/OU=I3S/CN=Franck Michel"
```

Keep in mind that if voms-proxy-info –all returns something like this:

```
attribute: /biomed/Role=NULL/Capability=NULL
```

Then the corresponding section in config.properties should be similar to:

```
HYDRA_CREATE_1=/biomed/Role=NULL/Capability=NULL
```

Run the configuration script:

```
./configure --withpass=_mysql_root_password_ --values /opt/glite/etc/glite-data-hydra-service/config.properties
```

Should one want to uninstall Hydra, just run the /opt/glite/etc/glite-data-hydra-service/unconfigure script: the DB and all instances will be removed.

An issue with the jakarta-commons-dbcp package must be handled: class BasicDataSourceFactory has moved from one java package to another in a recent version of the rpm. Therefore if the command below returns this result:

```
[root#] jar tf /usr/share/java/commons-dbcp-1.2.1.jar | grep BasicDataSourceFactory
org/apache/commons/dbcp/BasicDataSourceFactory.class
```

Finally manually update all files $CATALINA_HOME/conf/Catalina/localhost/*glite-data-hydra-service.xml as follows:

```
<Resource name="jdbc/hydra" auth="Container"
  (...)
  factory="org.apache.commons.dbcp.BasicDataSourceFactory"
```

```
  (...)
/>
```

5.2 Pre-installing Hydra client CLI on sites

5.2.1 Description

For users' jobs to be able to use the Hydra service, the glite-eds-* commands from the Hydra client command line interface (CLI) package must be pre-installed on sites. This pre-installation follows the *Shared Area Method* (see [R5]). It consists in deploying some VO-specific software, required in run time environment, on a shared space that all WNs of a given CE may access through the variable $VO_VONAME_SW_DIR. This process takes place as a specific job submission by the VO Software Manager.

For each site where the VO-specific software has been deployed, a runtime environment tag is published using the BDII GlueHostApplicationSoftwareRunTimeEnvironment attribute. This will help the match making process select those sites when a job specifically requires that software to be pre-installed.

The procedure below must be executed by a biomed VO Software Manager.

5.2.2 Software deployment

1. The binaries of the Hydra client CLI rpm have been repackaged as a simple tgz archive: glite-data-hydra-cli-3.1.2-4.sl5.x86_64.tgz. Register that file on an SE:

```
lcg-cr --vo biomed \
 -l lfn:/grid/biomed/glite-data-hydra-cli-3.1.2-4.sl5.x86_64.tgz \
 file:`pwd`/ glite-data-hydra-cli-3.1.2-4.sl5.x86_64.tgz
```

2. Prepare the deployment script hydra_client_cli_install.sh:

```
#!/bin/sh
# This script deploys the Hydra client CLI binaries to the shared area

# TAR distribution of the Hydra client CLI package
export BIN_ARC=glite-data-hydra-cli-3.1.2-4.sl5.x86_64.tgz

# Get a copy of the software
cd $VO_BIOMED_SW_DIR
lcg-cp --vo biomed \
 lfn:/grid/biomed/$BIN_ARC \
 file:`pwd`/$BIN_ARC

# Unzip it locally
tar zxf $BIN_ARC
rm -f $BIN_ARC
```

3. Then for each CE from a site where the Hydra client CLI software is to be published, do as follows:

3.1. Prepare the job description hydra_client_cli_install.jdl:

```
Executable    = " hydra_client_cli_install.sh";
StdOutput     = "std.out";
StdError      = "std.err";
InputSandbox  = {"hydra_client_cli_install.sh"};
OutputSandbox = {"std.out","std.err"};
Requirements  = RegExp(".*grid10.lal.in2p3.fr.*",other.GlueCEUniqueID);
```

3.2. To have right access to the shared area the job must be submitted with Software Manager authorizations.

```
$ voms-proxy-init --voms biomed:/biomed/Role=lcgadmin
```

Then submit the job normally.

5.2.3 Publication of the runtime environment tag

1. Prepare the script hydra_cli_tag.sh:

```
#!/bin/sh
# This script published a runtime environment tag for the Hydra client CLI

# Location of tag file and definition of tag
taglist=/opt/edg/var/info/biomed/biomed.list
tag=VO-biomed-hydra-client-cli-3.1.2-4.sl5.x86_64

# Publish software tag
touch ${taglist}
cp ${taglist} ${taglist}.bak
cat - ${taglist} <<EOF | sort -u > ${taglist}.new
${tag}
EOF
mv ${taglist}.new ${taglist}
```

2. Submit that script on each CE that is to be enabled to use the Hydra service, e.g.:

```
$ globus-job-run grid10.lal.in2p3.fr:2119/jobmanager-fork -s hydra_cli_tag.sh.sh
```

3. Verify that the tag has been published:

```
$ ldapsearch -LLL -H ldap://cclcgtopbdii02.in2p3.fr:2170 \
 -x -b 'mds-vo-name=local,o=grid' \
 '(GlueHostApplicationSoftwareRunTimeEnvironment=\
    VO-biomed-hydra-client-cli-3.1.2-4.sl5.x86_64)' \
 GlueChunkKey
```

5.3 Registration with the Service Discovery

The Hydra service must be registered in the BDII so that clients can use the Service Discovery configuration mode. This is a manual process done on request by a BDII administrator.

In the current experimental deployment, the service is configured using the local file mode. The registration with the Service Discovery shall be done only when the production configuration has been deployed.

6 APPENDIX - HYDRA CLIENT-SIDE

From the user's point of view, the service may be configured with two methods: a local-file based method, or through the service unique id published in the BDII. The latter is the preferred and most simple solution. For users' convenience, both solutions are described here after.

6.1 Client configuration using a local file

On each gLite UI, the following procedure must be applied.

As root, create or update file /opt/glite/etc/services.xml to reflect the different key repositories. This file can also be created as a regular user's file in their home directory for instance. A minimum file should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
  <service name="hydra-1">
    <parameters>
      <endpoint>https://egee1.unice.fr:8445/1glite-data-hydra-service/services/Hydra</endpoint>
      <type>org.glite.Metadata</type>
      <version>1.3.1</version>
      <volist><vo>biomed</vo></volist>
    </parameters>
    <associatedservices>
    </associatedservices>
  </service>
</services>
```

Set the following variables in the users' profile:

```
export GLITE_SD_PLUGIN=file
export GLITE_SD_SERVICES_XML=/opt/glite/etc/services.xml
```

6.2 Client configuration using the Service Discovery (BDII)

Set the following variables in the users' profile:

```
export GLITE_SD_PLUGIN=bdii
export GLITE_SD_VO=biomed
```

6.3 Client command-line interface

To use the Hydra service a user should first initialize a valid proxy certificate.

Create a key in Hydra, e.g. with id *fmichel-id*:

```
# glite-eds-key-register -v fmichel-id
A key has been generated and registered for ID 'fmichel-id'
```

Encrypt a local file:

```
# echo test > test.txt
# glite-eds-encrypt fmichel-id test.txt test.txt.encrypted
File 'test.txt' has been successfully encrypted
      with key 'fmichel-id'
      and written to 'test.txt.encrypted'.
```

Decrypt a local file:

```
# glite-eds-decrypt fmichel-id test.txt.encrypted text_decrypted.txt
File 'test.txt.encrypted' has been succesfully decrypted
      with key 'fmichel-id'
      and written to 'text_decrypted.txt'.
```