

EGEE

EGEE User's Guide

DGAS - HOME LOCATION REGISTER

Document identifier:	EGEE-JRA1-TEC-571271-HLR
Date:	November 7, 2005
Activity:	JRA1: Middleware Engineering and Integration
Lead Partner:	INFN
Document status:	DRAFT
Document link:	https://edms.cern.ch/document/571271/1

Abstract: This document describes the client/server architecture of the DGAS Home Location Register service, its command line utilities and C++ API, as well as its installation and configuration. Furthermore it explains how to manage the accounts of users and resources on an HLR server.

Delivery Slip

	Name	Partner	Date	Signature
From	A.Guarise, G. Patania, R.M.Piro	INFN Torino		
Reviewed by				
Approved by				

Document Change Log

Issue	Date	Comment	Author
Initial version	February 25,2005	-	A.Guarise, G. Patania, R.M.Piro
Updated version	August 5, 2005	updated Introduction, Installation and Configuration, added daemon start-up and new tools	A.Guarise, G. Patania, R.M.Piro
Updated version	September 23, 2005	minor improvements	A.Guarise, G. Patania, R.M.Piro
Updated version	November 7, 2005	added glite-dgas-hlr_updateDB.pl; name change: now Distributed Grid Accounting System; minor changes	A.Guarise, G. Patania, R.M.Piro

Document Change Record

Issue	Item	Reason for Change

Copyright ©Members of the EGEE Collaboration. 2005. See <http://eu-egee.org/partners> for details on the copyright holders.

EGEE (“Enabling Grids for E-science in Europe”) is a project funded by the European Union. For more information on the project, its partners and contributors please see <http://www.eu-egee.org>.

You are permitted to copy and distribute verbatim copies of this document containing this copyright notice, but modifying this document is not allowed. You are permitted to copy this document in whole or in part into other documents if you attach the following reference to the copied elements: “Copyright ©2005. Members of the EGEE Collaboration. <http://www.eu-egee.org>”

The information contained in this document represents the views of EGEE as of the date they are published. EGEE does not guarantee that any information contained herein is error-free, or up to date.

EGEE MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

CONTENTS

1	INTRODUCTION	5
1.1	HOME LOCATION REGISTER PURPOSE	5
1.1.1	USER AND RESOURCE ACCOUNTS	5
1.1.2	RESOURCE USAGE RECORDS	5
1.2	FEATURES	6
1.3	SERVICE ARCHITECTURE	7
1.3.1	HLR SERVER ENGINES	7
1.3.2	THE TRANSACTION MANAGER DAEMON	12
1.3.3	THE HAD DAEMON	12
1.4	INTERACTIONS WITH OTHER SERVICES	15
1.4.1	USER HLRS AND RESOURCE HLRS	15
1.4.2	DGAS USAGE RECORD ROUTING	15
2	QUICKSTART GUIDE	17
3	REFERENCE GUIDE	18
3.1	COMMANDLINE INTERFACES	18
3.1.1	HLR SERVICE CLIENTS	18
3.1.2	ADMINISTRATION TOOLS FOR ACCOUNT MANAGEMENT	27
3.1.3	ADMINISTRATION TOOLS FOR DATABASE QUERIES	38
3.1.4	ADMINISTRATION TOOLS FOR DATABASE CONVERSIONS	46
3.1.5	ADMINISTRATION TOOLS FOR DATABASE UPDATES	49
3.2	APPLICATION PROGRAM INTERFACES	50
3.2.1	PINGING THE HLR SERVER	50
3.2.2	CONTACTING THE JOBAUTHENGINE	51
3.2.3	SENDING USAGE RECORDS TO THE ATMENGINE	52
4	INSTALLATION	52
4.1	INSTALLATION OF THE HLR CLIENT PROGRAMS	52
4.2	INSTALLATION OF THE HLR SERVER	53
5	CONFIGURATION	54
5.1	PRELIMINARY CONFIGURATION	54
5.1.1	SECURITY ENVIRONMENT FOR THE HLR SERVER	54
5.1.2	SYNCHRONIZING THE HLR HOST TO A TIME SERVER	55
5.2	HLR CONFIGURATION FILE	55
5.2.1	DATABASE SPECIFIC PARAMETERS	56
5.2.2	SECURITY SPECIFIC PARAMETERS	57
5.2.3	DAEMON SPECIFIC PARAMETERS	57

5.2.4	LOG FILES	58
5.2.5	LOCK FILES	58
5.2.6	TRANSACTION QUEUE MANAGER SPECIFIC PARAMETERS	58
5.3	CONFIGURATION OF THE HLR DATABASE	59
5.3.1	CONFIGURATION OF THE MYSQL SERVER	59
5.3.2	CREATION OF THE HLR DATABASE	59
5.4	CONFIGURATION OF OPTIONAL ADMINISTRATION TOOLS	59
5.4.1	CONFIGURATION OF AUTOMATIC USER AND RESOURCE ACCOUNT IMPORT	60
5.4.2	CONFIGURATION OF CONVERSION OF ACCOUNTING RECORDS FROM DGAS TO APEL	60
6	RUNNING THE HLR SERVER	60
7	KNOWN PROBLEMS AND CAVEATS	61

1 INTRODUCTION

The Distributed Grid Accounting System (DGAS), previously called DataGrid Accounting System, aims to be a full featured distributed Grid accounting toolkit. Since it is conceived and designed to be completely grid oriented, it is fully distributed without having a central repository of accounting information. It instead relies upon a network of independent accounting servers used to keep the accounting/transaction records of groups of GridUsers and GridResources.

DGAS can be used to account classic Computational Usage Records like CPU Time, memory usage and so on. It can also be used as an Economic Accounting system, treating information about the cost of the jobs executed by each GridUser on the single GridResources. This feature can be exploited for example by a Grid Service Provider that wants to charge its users for the provided service. As described in [4, 5, 6] the economic accounting can also be used to implement the so called *Economic Brokering* of grid resources (selection of execution sites and services based on economic principles in order to improve the balancing of the workload).

1.1 HOME LOCATION REGISTER PURPOSE

The Home Location Register (HLR) service is the part of DGAS that is responsible for keeping the accounting information for both grid users and grid resources. It receives the accounting information, the so called *Usage Records* from the grid resources, and stores them for later retrieval. These usage records are the basis for the job cost computation¹, the phase in which the HLR computes the cost for a given job. The job cost can then be debited to the grid user and credited to the grid resource, thus implementing an economic accounting for the the grid activities of the single users. Information can be gathered from the HLR service on a per user, per resource, per job basis.

1.1.1 USER AND RESOURCE ACCOUNTS

The HLR associates accounting information to previously registered user and resource accounts. For more information on these accounts and their registration with the HLR server see Section 3.1.2.

1.1.2 RESOURCE USAGE RECORDS

Additionally to the general information for user and resource accounts the HLR server stores transaction/resource usage information for single jobs associated to the user and/or resource accounts. The following information is stored for each transaction/job usage record:

- user and/or resource account to which the record is associated.
- global job ID.
- global grid ID of the CE that executed the job.
- HLR contact string of the Resource HLR that received the transaction (if associated to a local user account).
- HLR contact string of the User HLR that sent the transaction (if associated to a local resource account).

¹Together with the prices furnished by the other core service of DGAS, the Price Authority. The prices are expressed in Grid Credits per Unit of Computational Energy (UCE) [4]. Currently the UCE is defined at the HLR-level as 100 seconds of CPU time, hence the total cost of a job is the CE's price, multiplied by the job's CPU time in seconds divided by 100.

- Job cost (in Grid Credits).
- price validity timestamp (in seconds since Jan. 1st, 1970).²
- transaction timestamp (in seconds since Jan. 1st, 1970).
- CPU and wall clock time in seconds.
- physical and virtual memory in KBytes.
- number of processors.
- creation time of the usage record (created by Giandua on the CE, see [3]).
- local LRMS job name on the CE.
- local user ID on the CE.
- local group ID on the CE.
- The user's VOMS FQAN (Fully Qualified Attribute Name).
- timestamp of job execution start (in seconds since Jan. 1st, 1970).
- timestamp of job execution stop (in seconds since Jan. 1st, 1970).
- timestamp of the job creation on the LRMS.
- timestamp of the job insertion into the last queue of the LRMS.
- the timestamp of the job insertion into an execution queue (eligible to run) of the LRMS.
- SpecInt2000 benchmark of the CE.
- SpecFloat2000 benchmark of the CE.
- timezone valid for the CE (allows to compute local times from the UTC timestamps).
- the VO for which the user submitted the job.

1.2 FEATURES

The current release of the DGAS Home Location Register module has the following features:

- Secure communication between client and server with encryption and mutual authentication implemented using *Globus GSI* [1].
- XML communication protocol between client and server.
- Persistent accounting information archive built upon a MySQL Relational Database.
- Both *standard* accounting info (like *cpuTime*, *Memory* etc...) and *economic* accounting info.
- Possibility to retrieve accounting info for each resource, user or job.
- User and resource Usage Records are available also as aggregate info.

²The job cost depends on the resource usage/"consumption" and the price that was valid on job submission time.

- Server ping service: it is possible to monitor the HLR server status from a remote site. This allows fast problem detection.
- CLI user interface, used to remotely retrieve accounting information from an HLR server.
- CLI HLR admin interface, used by the HLR administrator to manage the accounts.

1.3 SERVICE ARCHITECTURE

DGAS is designed as a web service, where a light client communicates a request using an XML-based protocol to a server. DGAS is composed of two complementary services, the HLR itself and the Price Authority (PA), described in [2]. Each request coming from a client is parsed by the responsible server and processed by a specific library module. Technically both the HLR and PA are based on the same web server³ that simply loads different modules implementing different functionalities. The information needed by the servers are kept in MySQL databases.

The main role of the HLR is to receive the Usage Records (URs) for each job executed by one of the Users whose accounts are managed by that HLR. These URs are then stored for later retrieval and used to compute the job cost⁴. The HLR is also responsible for managing the accounting transaction between the User HLR (that manages the account of the grid user), and the Resource HLR (that manages the account of the grid resource that executes the user's job). In Fig. 1 the UML Deployment diagram for a HLR server, a User Interface (UI) node and a Computing Element (CE) is shown.

1.3.1 HLR SERVER ENGINES

The Home Location Register server daemon listens for incoming TCP connections. When it receives a message from a client it loads the proper engine to process it (see Fig. 2). The HLR server currently implements the following engines:

- **pingEngine:** used to report the current status of the server in terms of connection per type received and the number of requests that caused an internal error.
- **jobAuthEngine:** receives from the Grid User, via the User interface, an authorization to treat the accounting information for a given job and to process the economic transactions originating from that job.
- **atmEngine:** receives the Usage Records of a job from the CE that executed it and retrieves the price assigned for that job⁵. It composes a transaction and queues it in a persistent transactionQueue for later (asynchronous) processing by a transaction manager daemon that is part of the HLR server.
- **urwgEngine:** has the same purpose as the *atmEngine*, but excepts usage records in the format defined by the GGF Usage Record Work Group (UR-WG).
- **bankEngine:** manages a crediting request (payment of the job cost to the resource's account) coming from the User HLR, that is the HLR server that manages the user's account.
- **uiEngine:** the server backend for the simple user interface requests. It allows a remote HLR User Interface to retrieve information about account status, transactions and so on.

³That is, two instances listening on different ports, one as PA server and one as HLR server.

⁴The job cost computation is based on the usage records and the resource prices provided by the PA service, see footnote of Section 1.1.

⁵by querying the PA server responsible for setting the price of the Computing Element in question.

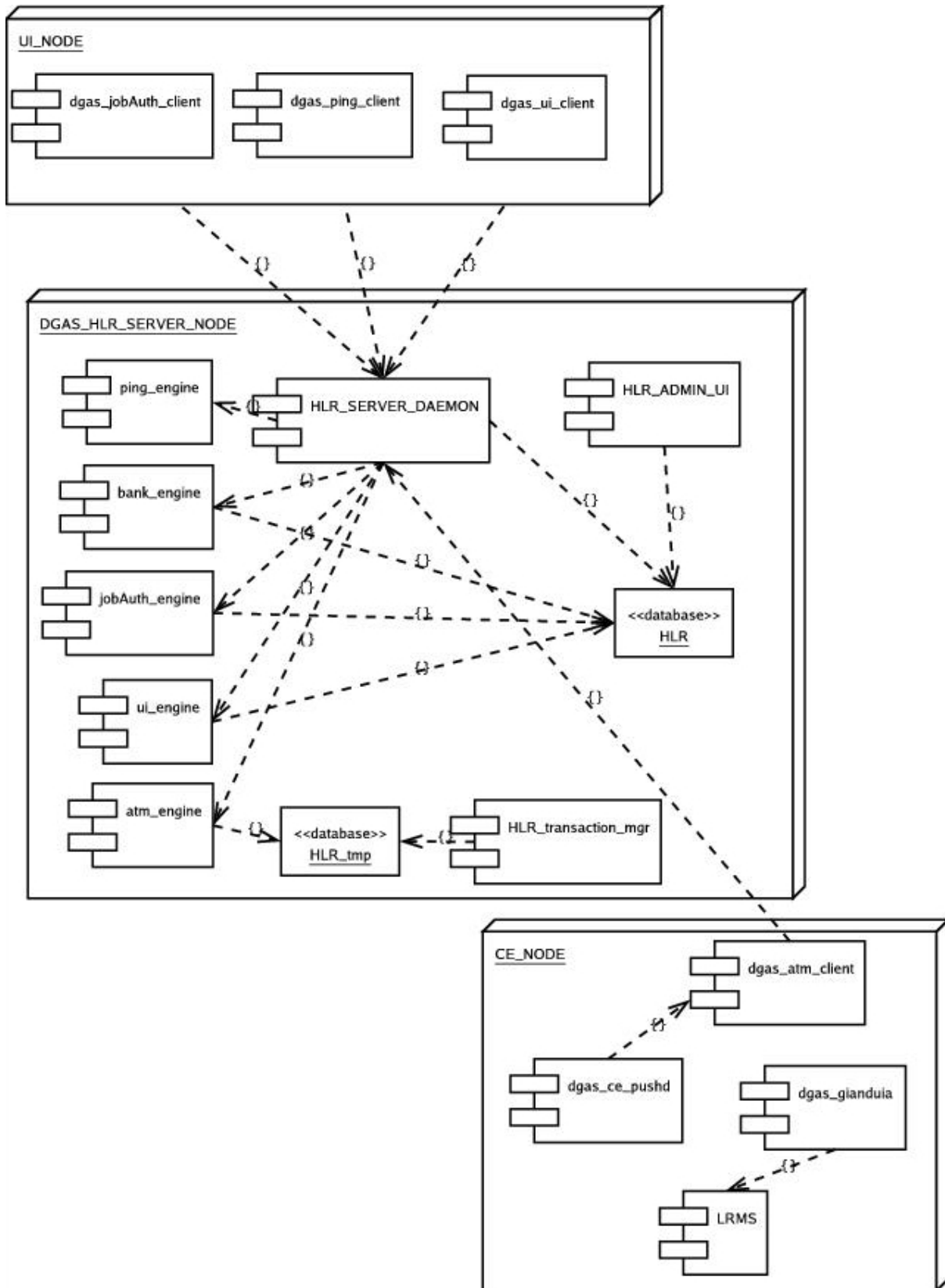


Figure 1: The UML deployment diagram for an HLR server, a generic UI node and a Computing Element.

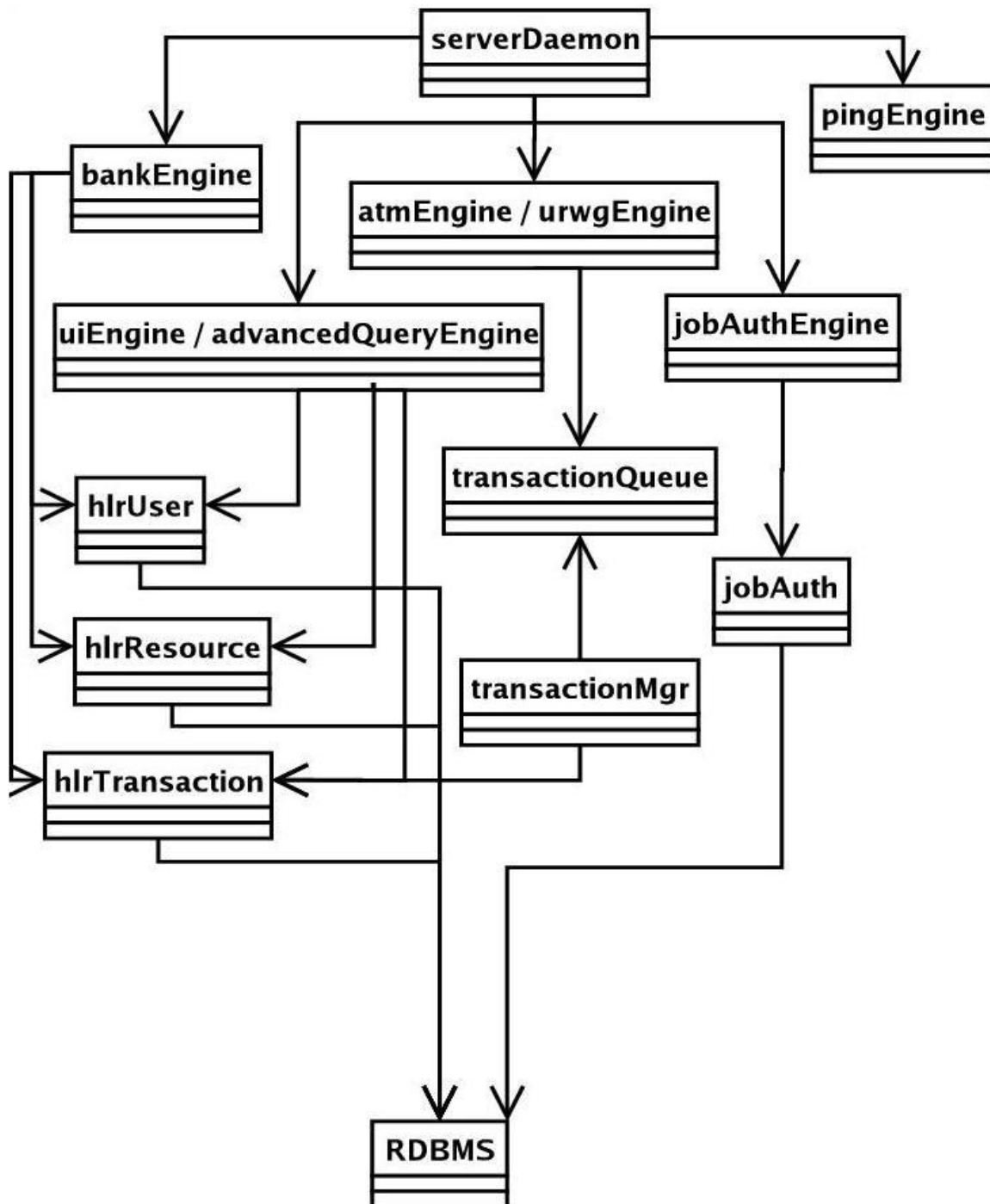


Figure 2: Architecture diagram for the Home Location Register Server

- **advancedQueryEngine:** has a similar purpose as the *uiEngine*, but composes more complex queries to the underlying database and is thus much more flexible (see the description of the `glite_dgas_hlrAdvancedQueryClient` in Section 3.1.1).

The diagrams in Fig. 1 and Fig. 2 also show a *transaction manager* that is used to retrieve the pending transactions from the *transaction queue* and asynchronously manage the crediting/debiting process.

The following are more detailed descriptions of the single HLR engines. The clients used to query the HLR server from remote sites are described in more detail in Section 3.1.1.

pingEngine This service allows the user or administrator to check if a given HLR (or PA) server is still alive or not. The working principle is rather simple: every time the server daemon receives an incoming request, it increments a counter for the engine involved. Every time an engine, processing a request, encounters an error such that the request can't be fulfilled, the engine increments an error counter. The ping engine simply reads these counters and returns the current state of the server to the ping client.

jobAuthEngine This service is notified by the UI, on behalf of the user, of the submission of a given job, such that the accounting process for that job is authorized with the user credentials. The HLR server that manages the user's account (User HLR) receives the `edg-jobId` of the job, the user X509 DN (implicitly received with the security context) and the timestamp⁶ of the job submission. These information are needed when processing the job accounting Usage Records and crediting/debiting the resource/user accounts. The User HLR that receives the request of processing the Usage Records for a job checks if a *job authorization* for that job exists. Only if that authorization exists the HLR can proceed, since it is sure that the job payment request has been authorized by the user, that implicitly signed the authorization with his X509 credentials.⁷ The deployment schema of the jobAuth service and its relationship with the gLite User Interface is shown in Fig. 3.

uiEngine This engine processes requests coming from the DGAS user interface. It simply processes a request with an user query by interrogating the underlying database to retrieve the needed info. It can answer to queries about grid users, resources, and about their jobs, reporting economic information such as the account status or the amount of grid credits exchanged in a single transaction⁸. The Usage Records of the jobs submitted by a user can also be retrieved as aggregate values, for example the total amount of *CPU time* consumed by a user, or as the detailed report of the resources consumed by each single job.

advancedQueryEngine The *advancedQueryEngine* is similar to the *uiEngine*, but allows to compose more complex queries to the underlying database and is thus much more flexible. For more information on advanced queries see the `glite_dgas_hlrAdvancedQueryClient` described Section 3.1.1.

ATMEngine This engine has the responsibility for receiving the Usage Records of the jobs submitted by the users that are part of the HLR's administrative domain. It is the Computing Element (CE) that, using the `glite_dgas_atmClient` upon job completion, sends the Usage Records to the User HLR. When the ATMEngine receives the Usage Records, it first checks that the User that submitted the job has a valid

⁶The timestamp of the job submission is needed when computing the job cost, since it is essential to compute the cost using the price of the Computing Element that was valid when the job has been submitted. Infact a long amount of time may occur between the job submission and its cost computation.

⁷Note that currently the advance authorization is not enforced, but in the future it will be required.

⁸Usually for each executed job an economic transaction between the Grid Consumer, that is the user, and the Grid Producer, usually a Computing Element, will be processed, involving both the HLR that manages the user's account and the HLR that manages the CE's account.

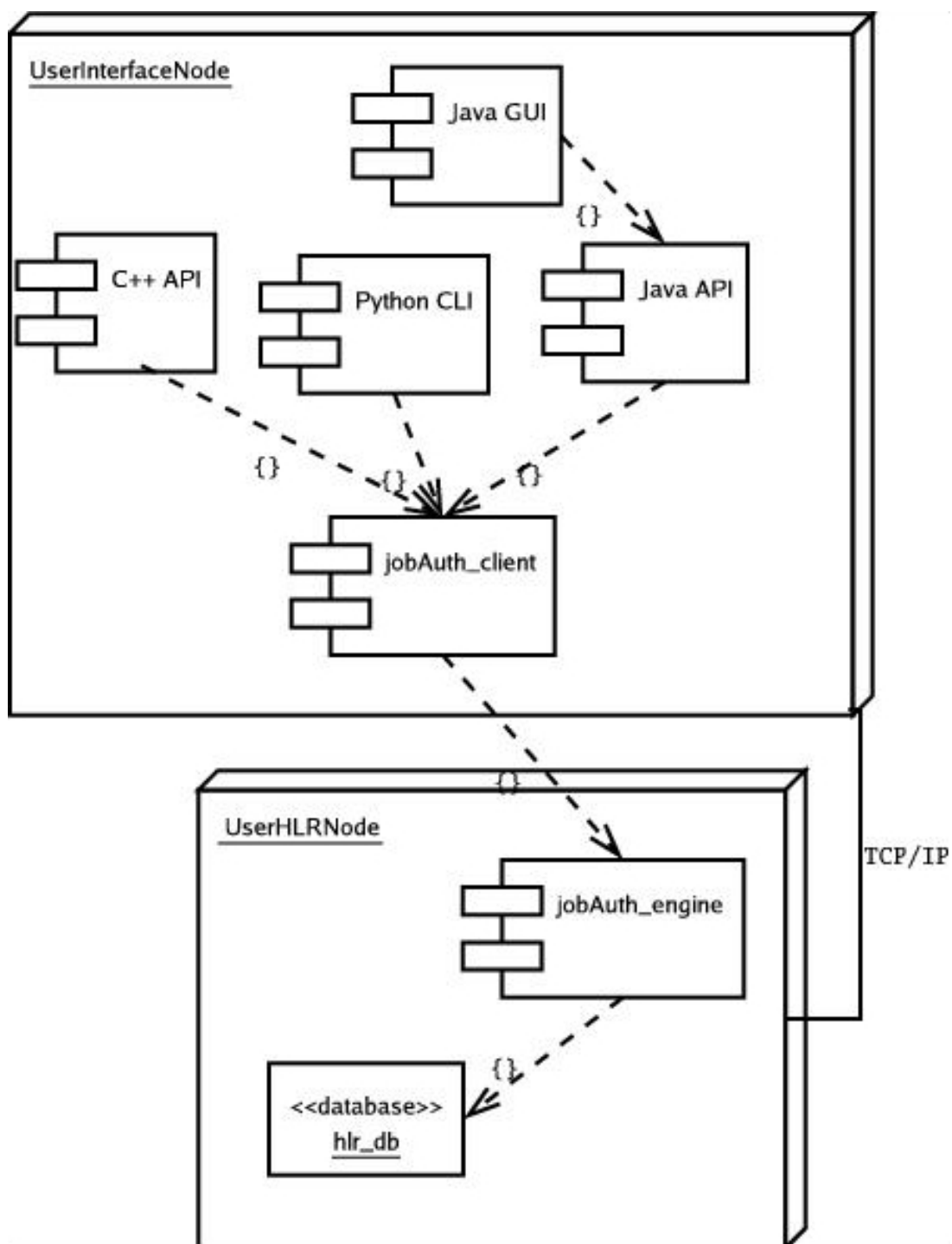


Figure 3: The UML deployment diagram for the Job Authorization service, There is one UI node, with the various bindings of the EDG/gLite User Interface, and the User HLR that receives the job authorization request.

account on the HLR and that the job has been successfully authorized (see Section 1.3.1). If everything is fine the Price Authority responsible for the CE is contacted to retrieve the price that was assigned to the CE at the time of job submission, then the transaction for the job is inserted into a persistent queue for asynchronous processing by a transaction manager daemon that is part of the HLR server. A conceptual view of the interaction between the Computing Element and the HLR using the ATM service is given as a UML sequence diagram in Fig. 4.

The deployment diagram of the ATM service is given in Fig. 5. The diagram shows a CE node and two HLR nodes, the User HLR that receives the Usage Records directly from the Computing Element, and the Resource HLR that is the end peer of the accounting transaction.

urwgEngine The *urwgEngine* is equivalent to the *ATMEngine* described above, but excepts usage records in the format defined by the GGF Usage Record Work Group (UR-WG). Although the GGF UR-WG format is now supported by DGAS the previous format as treated by the *ATMEngine* is still maintained for reasons of compatibility.

bankEngine This engine processes crediting requests from other HLRs, checking the existence of the account to be credited. If everything is fine the account is credited on the server side, and another account is debited on the client side⁹. A log of the processed transactions is kept on both HLRs involved in the transaction. The log can be retrieved using the DGAS user interface and contains information like the amount of grid credits exchanged, the job that originated the transaction, the usage records of the consumed resources, the price assigned to the resource for that job and the timestamp of the job submission.

1.3.2 THE TRANSACTION MANAGER DAEMON

The Transaction Manager is a daemon responsible for processing pending accounting transactions. Usage records are usually sent from the DGAS sensors installed on a Computing Element (see the DGAS Gianduia Guide [3]) to the User HLR (for more details on usage record routing see Section 1.4.2). The usage records are stored in a transaction queue in order to be processed by the Transaction Manager. Transaction processing usually includes the association of usage records to user/resource accounts as well as the communication between User and Resource HLR (that, in case economic accounting is configured, corresponds to a crediting/debiting process).

Since the Transaction Manager is automatically handled by the HLR start-up script (described in Section 6) the start-up of the daemon is not described here.

1.3.3 THE HAD DAEMON

Since DGAS treats important information, it has to provide a high availability. The *High Availability Daemon (HAD)* is responsible for continuously monitoring the status of a service. In case of failure it restarts the daemon, thus avoiding long down periods due to service failures.

This is implemented with a perl script that takes the filename of a daemon startup script as its only command line argument. This, however, is automatically handled by the HLR start-up script (described in Section 6) and thus the start-up of the HAD is not described here.

In order to be handled by HAD, the service startup script must understand the commands `status` and `restartMain`. The first one is the usual status command returning 0 if the daemon is alive and non-zero if it is stopped. The second one, `restartMain`, is a command that restarts all the processes managed

⁹The debiting is done by the User HLR that contacts the Resource HLR's *bankEngine* for the crediting/payment transaction.

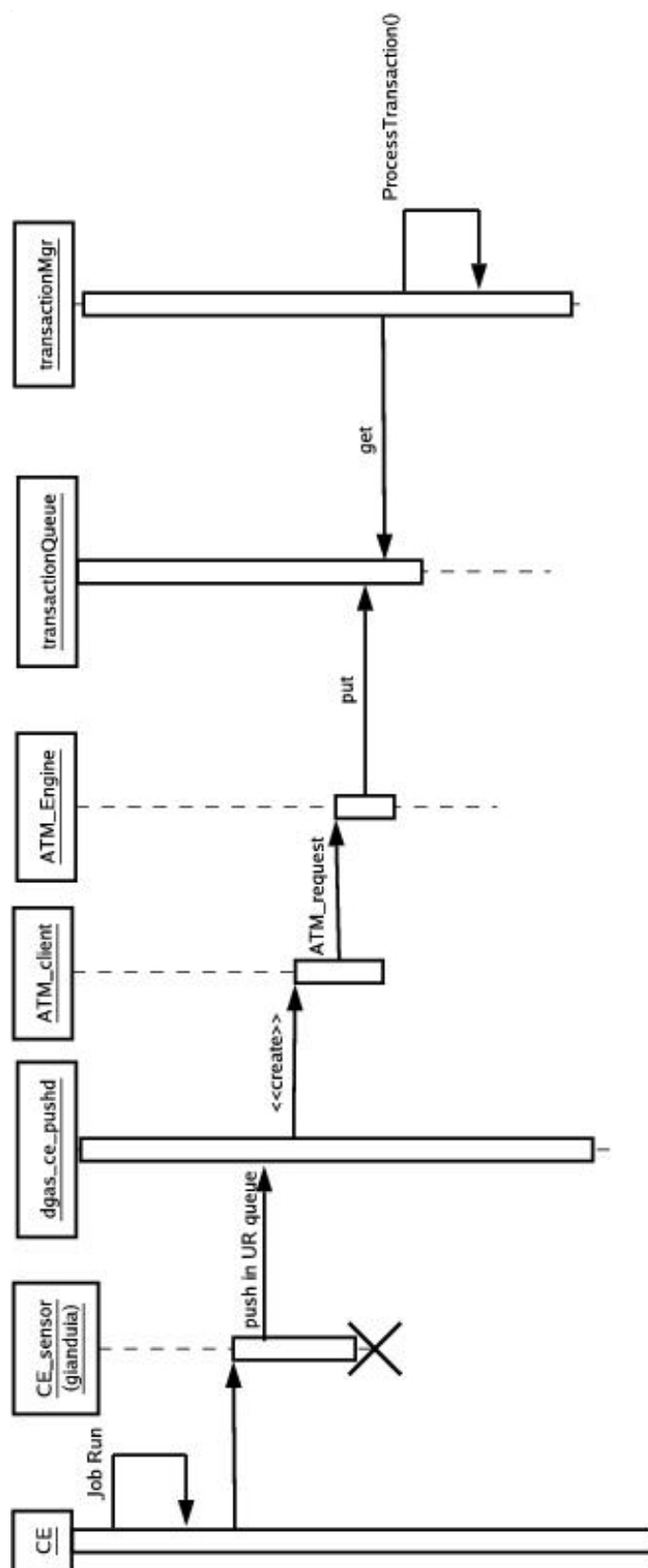


Figure 4: UML sequence diagram of the interaction between the CE and the HLR via the ATM service.

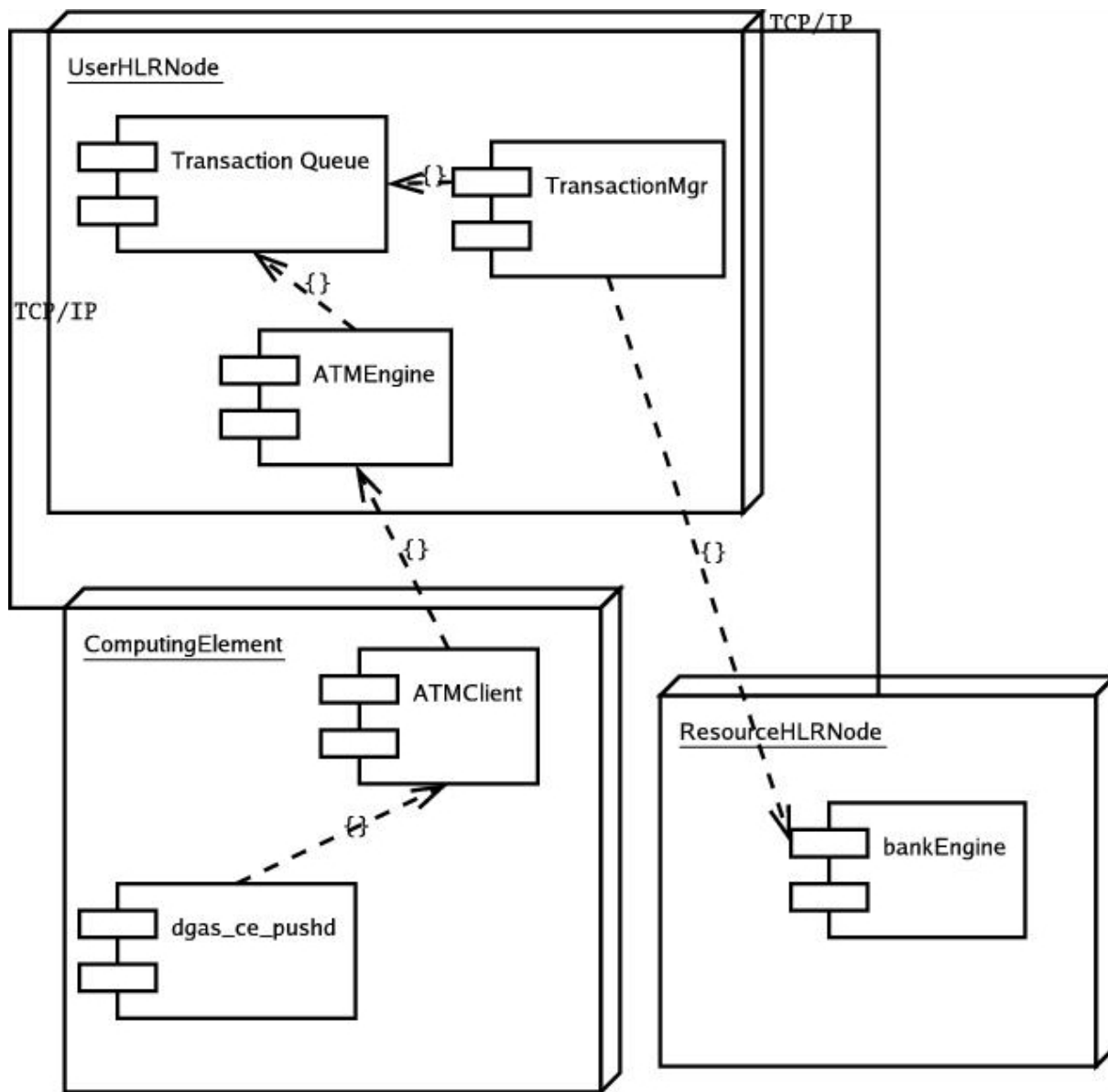


Figure 5: UML deployment diagram of the ATM service with one CE node, with the usage sensor and the ATM client module, and two HLR nodes. The User HLR receives the Usage Records from the CE node, while the Resource HLR is the end peer for the accounting transaction. Note: This diagram corresponds to the UDH-URR, see Section 1.4.2.

by the script apart from the HAD daemon itself. The HAD daemon is managed by the service scripts standard commands `start` and `stop` as well, thus `restartMain` is required as a command that restarts only the core services, but not the HAD; `restartMain` can then be used by the HAD to restart the core services, this avoids the HAD script to kill itself while attempting to restart the dead daemon.

1.4 INTERACTIONS WITH OTHER SERVICES

The HLR service interacts with several other services. As described in Section 1.3 the *jobAuthEngine* is contacted by the *User Interface* in order to authorize in advance the accounting (and payment) transactions for a submitted job.¹⁰ The *ATMEngine* is contacted by the DGAS *CE_Pushd* that is installed on the Computing Element and transmits the usage records collected by *Gianduia* [3]. The *bankEngine* is intended for communication between HLR servers for crediting/debiting transactions. The *uiEngine* finally provides information to arbitrary remote hosts (provided a valid user certificate is used, of course), replying to the commands described in Section 3.1.1.

Figure 6 shows a simplified sequence diagram of the default accounting (and pricing) process from job submission to resource usage accounting and crediting and debiting. As described in the following Sections the usage records can, however, also be routed in alternative ways.

1.4.1 USER HLRs AND RESOURCE HLRs

DGAS foresees two logical types of HLR servers.¹¹ The *User HLR* and the *Resource HLR*. A User HLR stores information from a user's point of view and is the place where users can ask for accounting information concerning themselves and the jobs they have submitted. A Resource HLR stores information from a site manager's point of view and is the place where site managers, or resource owners, can ask for information concerning their resources.

The reason of this division is straightforward. For scalability reasons there will be many User HLRs on the grid, and different users will be registered with different HLRs. Hence it is necessary that all accounting information concerning a given user be forwarded (and stored) on the HLR that manages his account ("User" HLR).

On the other side a CE will get job submissions from different users that will be registered with many different HLRs. It is clear that a resource owner that needs an exhaustive view of the CE usage can't query all the possible User HLRs to get all the information needed (the same, of course, is valid for users that shouldn't need to query several Resource HLRs). Hence copies of the usage records for all the job executed on a CE must be present on another HLR that manages the CE's account ("Resource" HLR). Thus a resource owner needs to query only a single HLR to have the exhaustive accounting view needed, preserving in this way a reasonable scalability.

1.4.2 DGAS USAGE RECORD ROUTING

There are currently three scenarios for routing the information among the HLRs:

1) *User-signed Dual HLR UR Routing (UDH-URR)*

CE $\xRightarrow{\text{(user credentials)}}$ **User HLR** $\xRightarrow{\text{(User HLR credentials)}}$ **Resource HLR**

This is the default information routing. The usage record, collected on the CE, is signed with the user's credentials and sent over a secure channel to the User HLR. From there the UR is forwarded, signed with the User HLR's credentials to the Resource HLR. The process is a double commit. Since usage records are signed with the users' credentials, economic accounting can be asked to the system.

¹⁰Note that currently the advance authorization is not enforced, but in the future it will be required.

¹¹An HLR server can, however, manage both user and resource accounts if required.

II) Resource-signed Dual HLR UR Routing (RDH-URR)

CE =(CE credentials) \implies **Resource HLR** =(Resource HLR credentials) \implies **User HLR**

This is an alternate route that can be used in two situations:

- The user's proxy certificate coming with the job is expired when the CE attempts to establish the communication with the User HLR (UDH-URR cannot be initiated);
- The CE's system administrator explicitly asks for this route.

In this scenario the job usage record is sent, signed with the CE credentials, over a secure channel to the Resource HLR. Then with the same double commit process, it is forwarded, signed with the Resource HLR credentials, to the User HLR. Since the UR is not signed with the user's credentials, economic accounting should be avoided.

III) Resource-signed Single HLR UR Routing (RSH-URR)

CE =(CE credentials) \implies **Resource HLR**

In this scenario, the usage records are sent only to the Resource HLR. This is a requirement for some sites that do not want that their accounting information leaves their administrative domain. No information will be available to the User HLR concerning jobs executed on such sites. Economic accounting, through the exchange of virtual credits between the user's and the resource's account, is not possible.

2 QUICKSTART GUIDE

Since the HLR service is responsible for an accurate accounting of the resource usage of single grid users, groups of users and entire Virtual Organizations (VOs), it is essential to install and configure it properly, especially if the information it provides is used in an economic context in which grid users pay virtual credits for the resources they use. We therefore recommend to read the following Sections carefully (especially the Sections on installation, configuration and administration), although we first make some examples for some of the most common use cases to give you an impression of the service furnished by an HLR server.

Example 1: A grid user that has an account on a particular HLR (most probably the HLR of his Virtual Organization¹²) and has the HLR clients installed on the local machine can easily query the remote HLR server for the status of the own account¹³:

```

> $GLITE_LOCATION/bin/glite_dgas_hlrUserInfoClient \
  -s hlr02.to.infn.it:56568: -c "/C=IT/O=INFN/OU=Personal \
Certificate/L=Torino/CN=testuser1/Email=testuser@to.infn.it"

|testuser1|testuser@to.infn.it|description of testuser1|/C=IT/O=INFN/OU=Personal
Certificate/L=Torino/CN=testuser1/Email=testuser@to.infn.it|testgroup1|10000|234|
1|wall_time=0,cpu_time=0,job_number=0|
  
```

where "hlr02.to.infn.it:56568:" is the contact string (see Section 3.1.1) of the HLR on which resides the user's account and -c specifies the user's certificate subject (DN). The HLR server's response (described more detailed in Section 3.1.1) contains information on the number of jobs submitted by the

¹²We propose to have one HLR server per VO.

¹³Since the user's certificate defines the security context, the user will have access ONLY to information on the own account. This of course implies that the user needs a valid proxy certificate, otherwise the connection to the HLR server cannot be established.

user and the CPU time and wall clock time “consumed” by the jobs. It additionally provides information on the funds (that is, virtual credits) assigned to the user, the funds that have been booked and the funds that have been spent for executed jobs. This information will be important in the context of a grid economy.

Example 2: Adding a new HLR account for a grid user. This operation is done locally by the HLR admin. The simplest way of creating a new user account is to use the command

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAddUser -i
```

that interactively asks the administrator for the necessary information. In order to allow the client to be used in automated scripts as well, the information may also be specified as commandline arguments (see Section 3.1.2 for more details).

The creation of resource accounts is quite similar (see Section 3.1.2). For more information, please read the reference guide in the following Section.

3 REFERENCE GUIDE

3.1 COMMANDLINE INTERFACES

In this section we describe the service clients used to contact the HLR engines (see Section 1.3.1), as well as the administrative commandline tools that allow to manage the user and resource accounts on the HLR server. It does however not describe client programs for all the HLR engines provided by the HLR server (see Section 1.3.1). The client programs installed on the Computing Element (such as the clients for the ATMEngine) that are necessary to communicate the usage record of an executed job to an HLR server are described in [3], since they are installed with and used by the usage records collecting and transmitting daemons *Gianduia* and *CE_Pushd*.

Note: The paths of the client programs are described using the environment variable `GLITE_LOCATION` that may be set during installation (the default value is `/opt/glite/`).

3.1.1 HLR SERVICE CLIENTS

The service clients described here are commandline programs that allow to query the HLR server from remote sites.

The remote HLR server is usually specified by a *server contact string* that has the following form:

```
" [HOSTNAME] : [PORT] : [CERT_SUBJECT] "
```

where `[HOSTNAME]`, `[PORT]` and `[CERT_SUBJECT]` are the server's hostname, the port on which it is listening and the subject of its certificate respectively. For example:

```
-s "h1r01.to.infn.it:56568:/C=IT/O=INFN/OU=DGAS_HLR/L=Torino\
/CN=h1r01.to.infn.it/Email=andrea.guarise@to.infn.it"
```

or, if mutual authentication between the client and the server is not required (don't forget the colon at the end of the string!):

```
-s "h1r01.to.infn.it:56568:"
```

glite_dgas_pingClient This is the client used to *ping* the server status. It can be used to easily check if the server is alive and responsive, or gather more detailed information about its current state.

The client program can be usually found at:

```
$GLITE_LOCATION/libexec/glite_dgas_pingClient
```

and has the following options:

```
> $GLITE_LOCATION/libexec/glite_dgas_pingClient -h
DGAS ping client
Author: Andrea Guarise <andrea.guarise@to.infn.it>
```

Usage:

```
glite_dgas_pingClient <OPTIONS>
```

Where options are:

```

-h --help                Display this help and exit.
-s --server <HLR/PA contact> Contact string of HLR or PA server.
                          The HLR contact string has the form:
                          "host:port:host_cert_subject"
-t --type <ping type>    Ping Type: "0" = Normal, "1" = Status info

```

The specification of the *server* is mandatory (see Section 3.1.1 for how to form the server contact string).

If launched with only the `-s` option the client simply checks if the given HLR (or PA) server is alive. Otherwise, if you specify the `-t 1` option you will get an output like this:

Server status:

```

Available Engines: UI:ATM:BANK:JOBAUTH:USERAUTH:PING
ATM requests: 1539/23
jobAuth requests: 36/0
Ping requests: 1

```

from which it is possible to see the server available engines, and the number of requests received (and error states encountered) per engine.

glite_dgas_hlrUserInfoClient This client can be used to query an HLR server for information about a *specific user* (for an *entire group of users* use the `glite_dgas_hlrAdvancedQueryClient`, see Section 3.1.1). The client program can be usually found at:

```
$GLITE_LOCATION/bin/glite_dgas_hlrUserInfoClient
```

and has the following options:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrUserInfoClient -h

glite_dgas_hlrUserInfoClient 1.0
Author: Andrea Guarise
```

Usage:

```

glite_dgas_hlrUserInfoClient [OPTIONS]
  -h --help                Display this help and exit
  -s --server <HLR contact> The contact string of the HLR to query.
                           The HLR contact string is of the form:
                           "host:port:host_cert_subject"
  -c --cert <cert_subject> The user's certificate subject
  -u --uid <userID>        The user's ID in the HLR database
  
```

Note: Exactly one of the options `-u` and `-c` has to be specified to identify the specific user.

The specification of the *HLR contact string* is mandatory (see Section 3.1.1 for how to form the contact string). The other parameters determine the user for which information is requested. Exactly one of the parameters `-u` and `-c` has to be specified.

IMPORTANT NOTE: When the client is used, your personal certificate will determine whether you have access to the requested information or not. If not, the client program will quit without complaining. The same will happen if the server does not find a matching record. For a simple user it is therefore not possible to find out which other users or resources are managed by the queried HLR server.

Example: Getting the status of your own account (your personal certificate has to be valid and match the certificate subject associated to your account on the HLR database).

```

> $GLITE_LOCATION/bin/glite_dgas_hlrUserInfoClient \
  -s hlr02.to.infn.it:56568: -u testuser1
  
```

The desired information will be displayed in a single line such as

```

|testuser1|testuser@to.infn.it|description of testuser1|/C=IT/O=INFN/OU=Personal
Certificate/L=Torino/CN=testuser1/Email=testuser@to.infn.it|testgroup1|10000|234|
1|wall_time=0,cpu_time=0,job_number=0|
  
```

The single fields are separated by a pipe (|) and have the following meaning:

- user id (here: testuser1)
- email (here: testuser@to.infn.it)
- description of user (here: description of testuser1)
- certificate subject (here: /C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=testuser1/Email=testuser@to.infn.it)
- group id (here: testgroup1)
- assigned grid credits (here: 10000)
- booked grid credits (here: 234)
- spent grid credits (here: 1)
- total wall clock time, CPU time and number of executed jobs (here: wall_time=0,cpu_time=0,job_number=0)

glite_dgas_hlrResourceInfoClient This client can be used to query an HLR server for *information about a specific resource* (for an *group of resources* use the `glite_dgas_hlrAdvancedQueryClient`, see Section 3.1.1). The client program can be usually found at:

```
$GLITE_LOCATION/bin/glite_dgas_hlrResourceInfoClient
```

and has the following options:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrResourceInfoClient -h
```

```
glite_dgas_hlrResourceInfoClient 1.0
Author: Andrea Guarise
```

Usage:

```
glite_dgas_hlrResourceInfoClient [OPTIONS]
  -h --help                Display this help and exit
  -s --server <HLR contact> The contact string of the HLR to query.
                           The HLR contact string is of the form:
                           "host:port:host_cert_subject"
  -c --ceId <ceID>        The global resource ID (CE ID)
  -r --rid <resID>        The resource's ID in the HLR database
```

The specification of the *HLR contact string* is mandatory (see Section 3.1.1 for how to form the contact string). The additional parameters `-c` and `-r` determine the resource for which information is requested (as global CE ID or local HLR account ID, respectively). It is enough to specify one of them.

IMPORTANT NOTE: When the client is used, your personal certificate will determine whether you have access to the requested information or not. If not, the client program will quit without complaining. The same will happen if the server does not find a matching record. Only HLR administrators are authorized to retrieve the information.

Example: Getting the status of a specific resource (your personal certificate has to be valid and match the certificate subject associated to an HLR or resource administrator on the HLR database).

```
> $GLITE_LOCATION/bin/glite_dgas_hlrResourceInfoClient \
  -s hlr02.to.infn.it:56568: -r grid002long
```

The desired information will be displayed in a single line such as

```
|grid002long|grid002.to.infn.it:2119/jobmanager-pbs-long|grid002.to.infn.it:2119/jobmanager-pbs-long|infngrid|58529|wall_time=406560,cpu_time=204960,job_number=56|
```

The single fields are separated by a pipe (|) and have the following meaning:

- resource id (here: `grid002long`)
- description of resource
(here: `grid002.to.infn.it:2119/jobmanager-pbs-long`)
- global CE ID
(here: `grid002.to.infn.it:2119/jobmanager-pbs-long`)

- group id (here: infngrid)
- total grid credits earned (here: 58529)
- total wall clock time, CPU time and number of executed jobs
(here: wall_time=406560, cpu_time=204960, job_number=56)

glite_dgas_hlrAdvancedQueryClient This client can be used for more advanced queries to retrieve lists of transactions as well as aggregated information on users, resources, account groups and VOs.

Note: It is important to keep in mind that the authorization to access the accounting data will be checked by the HLR server, hence the use of the client requires a valid user proxy certificate. Normal grid users can retrieve only information related to their own HLR account. Only users that are mapped as HLR administrators can access information regarding all user and resource accounts.

The client program can be usually found at:

```
$GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient
```

and has the following options:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient -h
```

```
./glite_dgas_hlrAdvancedQueryClient
Author: Andrea Guarise
13/09/2005
```

usage:

```
./glite_dgas_hlrAdvancedQueryClient [OPTIONS]
-H --hlrLocation <hlrContact> Specifies the contact string of the HLR to query.
                               The HLR contact string is of the form: "host:port:host_cert_subject"
-Q --queryType <queryType>   Specifies the type of query
-e --email <emailAddress>     Specifies the email address of the user
-u --userCert <cert_subject> Used to specify the user's certificate subject
-r --resourceId <ceID>        Used to specify the global CE ID for the resource
-v --voId <voID>             Used to specify the virtual organization
-g --vomsGroup <vomsGroup>   Restrict the queries for users to a given voms group
-t --time <interval>         Time interval to use in the query, in the form:
                               "<startTimeStamp>-<endTimeStamp>" : all the info in the given period
                               "-<endTimeStamp>" : all the info available until endTimeStamp
                               "<startTimeStamp>-" : all the info available from startTimeStamp
                               Timestamps are specified in seconds since Jan. 1, 1970 0:00:00 GMT
                               If not specified or '*': no filtering for the time period.
-F --Frequency <freq>        Frequency for the aggregation of query results
-D --debug                    Ask for debug output
-E --Extended                 Show more info
-j --jobId <jobID>           Select a specific global job ID
-h --help                     This help
```

The parameter <queryType> can be one of the following:

"userAggregate": Aggregate usage records for a user
 "resourceAggregate": Aggregate usage records for a resource
 "groupAggregate": Aggregate usage records for a group of accounts
 "voAggregate": Aggregate usage records for the users belonging to a VO
 "userJobList": List of jobs submitted by a user
 "resourceJobList": List of jobs executed by a resource

The parameter <freq> can be one of the following:

"day": daily aggregate usage records
 "week": weekly aggregate usage records
 "month": monthly aggregate usage records
 "hour": hourly aggregate usage records

The specification of the *servername* (`--hlrLocation`) is mandatory (see Section 3.1.1 for how to form the server contact string). The `--queryType` option as well is mandatory and can have one of the following arguments:

- `userAggregate`: aggregates information on the transactions related user accounts.
- `resourceAggregate`: aggregates information on the transactions related resource accounts.
- `groupAggregate`: aggregates information on the transactions related to the users belonging to a specific VOMS group (to specify with `--vomsGroup`).
- `voAggregate`: aggregates information on the transactions related to the users belonging to a specific VO (to specify with `--voId`).
- `userJobList`: retrieves detailed information on transactions related to user accounts.
- `resourceJobList`: retrieves detailed information on transactions related to resource accounts.

The parameter `--Frequency` can be used to divide the aggregated information into distinct periods (hours, days, weeks and months) instead of aggregating it into one single response (see Example 1 below).

For queries related to user accounts (queries of type `userAggregate`, `groupAggregate`, `voAggregate` and `userJobList`) the parameters `--email`, `--userCert`, `--vomsGroup`, `--voId` and `--jobId` can be used to restrict the transactions to be considered for the result.

For queries related to resource accounts (`resourceAggregate` and `resourceJobList`) the parameters `--resourceId` and `--jobId` can be used to restrict the transactions to be considered for the result.

Note: The parameters `--userCert` and `--resourceId` may contain SQL-compliant wildecards (that is '%' for an arbitray string and '_' for a single character, e.g. "%cern.ch%" for all Computing Elements at CERN as used below in Example 2).

The results returned for queries related to both user and resource accounts can be further restricted by using the parameter `--time` to specify an interval of timestamps (seconds since Jan. 1st, 1970) for the transactions. It is possible to specify only the lower or upper bound of the time interval (see `--help`).

The parameter `--Extended` allows to print more detailed transaction information (only for `userJobList` and `resourceJobList`) and the parameter `--debug` allows to add some information regarding the messages send to and retrieved from the HLR server.

The following are some simple examples that show how to form the most relevant queries as well as the results that can be expected:

Example 1: Getting aggregated information for a particular user, time-period and divided in days:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient \
  -H "h1r02.to.infn.it:56568:" -Q userAggregate \
  -u "/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Andrea \
  Guarise/Email=Andrea.Guarise@to.infn.it" \
  -t 1117164757-1118164757 -F day
```

The desired information will be displayed in one or more lines such as

	date	jobs	totCpuTime	totWallTime	totMem (MB)	totVMem (MB)
*	27/5/2005-28/5/2005	0	0	0	0	0
*	28/5/2005-29/5/2005	0	0	0	0	0
*	29/5/2005-30/5/2005	0	0	0	0	0
*	30/5/2005-31/5/2005	61	779	1041	109	415
*	31/5/2005-1/6/2005	70	933	1222	176	686
*	1/6/2005-2/6/2005	177	2317	3108	398	1608
*	2/6/2005-	363	4850	6079	865	3417

Example 2: Getting aggregated information for jobs executed by a the Computing Elements at a specific site:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient \
  -H "h1r02.to.infn.it:56568:" -Q resourceAggregate -r "%cern.ch%" \
  -t 1121149757-1121164757 -F hour
```

The desired information will be displayed in one or more lines such as

	date	jobs	totCpuTime	totWallTime	totMem (MB)	totVMem (MB)
*	8:30:12/7/2005-9:30:12/7/2005	16	27	101	19	82
*	9:30:12/7/2005-10:30:12/7/2005	14	22	87	33	131
*	10:30:12/7/2005-11:30:12/7/2005	15	25	102	20	76
*	11:30:12/7/2005-12:30:12/7/2005	32	72	294	69	244
*	12:30:12/7/2005-13:30:12/7/2005	10	18	84	24	100
*	13:30:12/7/2005-	0	0	0	0	0

Example 3: Getting a detailed list of jobs executed by a specific resource:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient \
  -H "h1r02.to.infn.it:56568:" -Q resourceJobList \
  -r "lxb2077.cern.ch:2119/blah-pbs-long" -t 1121163757-
```

The desired information will be displayed in one or more lines such as

```
*1|edgJobId=https://gundam.cnaf.infn.it:9000/UrKp8cPblz81b16v_eZHzQ\
|lxb2077.cern.ch:2119/blah-pbs-long|proto|/C=IT/O=INFN/OU=Personal \
Certificate/L=Padova/CN=Alessio Gianelle/Email=alessio.gianelle@pd.infn.it\
|local|Cost=0|cePriceTime=1121164514|cpuTime=2|wallTime=7|mem=0|vmem=0\
|userVo=egee
*2|edgJobId=https://gundam.cnaf.infn.it:9000/G5e6B8RYeOFFsD6eou-1mA\
|lxb2077.cern.ch:2119/blah-pbs-long|proto|/C=IT/O=INFN/OU=Personal \
Certificate/L=Padova/CN=Alessio Gianelle/Email=alessio.gianelle@pd.infn.it\
|local|Cost=0|cePriceTime=1121163783|cpuTime=2|wallTime=18|mem=7464|vmem=21980\
|userVo=egee
```

The single lines begin with an asterisk (*). The fields are separated by a pipe (|) and have the following meaning:

- item number (here: 1),
- job id (here: edgJobId=https ...),
- global grid ID of the executing resource (here: lxb2077 ...),
- the group to which the resource belongs (here: proto),
- x509 certificate subject of the user that submitted the job (here: /C=IT/O=INFN ...),
- the contact string of the user's HLR server (here: local, i.e. the user's account is on the same HLR server),
- the cost (in grid credits) of the job (here: Cost=0; see [2] for more information on pricing),
- the timestamp that indicates which price is valid for the usage record (usually the submission time of the job, here: cePriceTime=...; see [2] for more information on pricing),
- the CPU time consumed by the job in seconds (here: cpuTime=...),
- the wall clock time consumed by the job in seconds (here: wallTime=...),
- the physical memory (in KBytes) used by the job (here: mem=...),
- the virtual memory (in KBytes) used by the job (here: vmem=...),
- the VO for which the user submitted the job (here: userVo=).

Example 4: Getting a detailed list with **extended** information for jobs executed by a specific resource:

```
> $GLITE_LOCATION/bin/glite_dgas_hlrAdvancedQueryClient \
-H "h1r02.to.infn.it:56568:" -Q resourceJobList \
-r "lxb2077.cern.ch:2119/blah-pbs-long" -t 1121163757- -E
```

The desired information will be displayed in one or more lines such as

```
*1|edgJobId=https://gundam.cnaf.infn.it:9000/UrKp8cPblz81bl6v_eZHzQ\
|lxb2077.cern.ch:2119/blah-pbs-long|proto|/C=IT/O=INFN/OU=Personal \
Certificate/L=Padova/CN=Alessio Gianelle/Email=alessio.gianelle@pd.infn.it\
|local|Cost=0|cePriceTime=1121164514|cpuTime=2|wallTime=7|mem=0|vmem=0\
|urCreation=Tue Jul 12 12:35:23 2005|processors=1|localUserId=egEE005|lrmsId=lrmsId\
|jobName=blahjob_b21953|start=1121162718|end=1121162725|ctime=1121162676\
|qtime=1121162676|etime=1121162676|fqan=|si2k=400|sf2k=380|tz=+0200|userVo=egEE
*2|edgJobId=https://gundam.cnaf.infn.it:9000/G5e6B8RYeOFfSd6eou-1mA\
|lxb2077.cern.ch:2119/blah-pbs-long|proto|/C=IT/O=INFN/OU=Personal \
Certificate/L=Padova/CN=Alessio Gianelle/Email=alessio.gianelle@pd.infn.it\
|local|Cost=0|cePriceTime=1121163783|cpuTime=2|wallTime=18|mem=7464|vmem=21980\
|urCreation=Tue Jul 12 12:23:28 2005|processors=1|localUserId=egEE005|lrmsId=lrmsId\
|jobName=blahjob_x20955|start=1121162659|end=1121162677|ctime=1121162649\
|qtime=1121162649|etime=1121162649|fqan=|si2k=400|sf2k=380|tz=+0200|userVo=egEE
```

The single lines begin with an asterisk (*). The fields of the **extended** output are separated by a pipe (|) and have the following meaning:

- item number (here: 1),
- job id (here: edgJobId=https ...),
- grid ID of the executing resource (here: lxb2077 ...),
- the group to which the resource belongs (here: proto),
- x509 certificate subject of the user that submitted the job (here: /C=IT/O=INFN ...),
- contact string of the user's HLR server (here: local, i.e. the user's account is on the same HLR server),
- cost (in grid credits) of the job (here: Cost=0; see [2] for more information on pricing),
- timestamp that indicates which price is valid for the usage record (usually submission time of the job, here: cePriceTime=...; see [2] for more information on pricing),
- CPU time consumed by the job in seconds (here: cpuTime=...),
- wall clock time consumed by the job in seconds (here: wallTime=...),
- physical memory (in KBytes) used by the job (here: mem=...),
- virtual memory (in KBytes) used by the job (here: vmem=...),
- creation time of the usage record (here urCreation=...),
- number of processors used by the job (here processors=...),
- local user id on the CE (here localUserId=...),
- LRMS id (here lrmsId=...),
- local LRMS job name (here jobName=...),
- timestamp of the execution start (here start=...),
- timestamp of the execution stop (here stop=...),
- timestamp of the job creation on the LRMS (here ctime=...),

- timestamp of the job insertion into the last queue of the LRMS (here `qtime=...`),
- the timestamp of the job insertion into an execution queue (eligible to run) of the LRMS (here `qtime=...`),
- the VOMS Fully Qualified Attribute Name (FQAN) of the user (if available from the user proxy certificate; here `fqan=...`),
- SpecInt2000 benchmark of the CE (here `si2k=...`),
- SpecFloat2000 benchmark of the CE (here `sf2k=...`),
- timezone valid for the CE (allows to compute local times from the UTC timestamps; here `tz=...`),
- the VO for which the user submitted the job (here: `userVo=...`).

3.1.2 ADMINISTRATION TOOLS FOR ACCOUNT MANAGEMENT

Both grid users and resources need to be registered on an HLR in order for their grid activities to be accounted. With the “registration” an “account” is created on the HLR. This account is a repository of information concerning the Grid User or Resource. The information available is:

Grid User

- User Name: The user's real name.
- E-Mail: The user's email address.
- Description: An arbitrary string describing the user. Often the user's certificate subject is used instead.
- Certificate Subject: The user's Certificate Subject used to register the account.
- Group ID : The ID of the Group to which the user belongs (e.g. “infngrid”).
- VO ID: The ID addressing the Virtual Organization to which the user belongs (e.g. “INFN”). It is important to note that although each account in the HLR database is associated to only one VO, the transaction information/usage records stored in the HLR database keep track of the VO for which a job was submitted even if it is different from the VO to which the user account is associated on the given HLR server.¹⁴
- Assigned Funds: The amount of grid credits assigned to the user for grid activities. Currently it doesn't really impose a limit for grid resource usage, but in the future it will eventually be used to impose usage quotas to users when needed.
- Booked Funds: Amount of grid credits booked for the current grid activities of the user (e.g. for pending or running jobs).
- Spent Funds: Amount of grid credits spent by the user for his past grid activities (Note: the amount spent for single jobs, that is the job cost, will be stored in the transaction information associated with the jobs).

¹⁴This enhances flexibility since it allows users that belong to multiple VOs to either have multiple accounts on multiple HLRs (we recommend one HLR per VO) or have a single account that keeps track of jobs submitted on behalf of more than one VO.

- **CPU Time:** The amount of CPU time consumed by the user's grid applications. Many levels of aggregation are possible: for single jobs, for all the jobs, for all the jobs submitted in a given time period.
- **Wall Clock Time:** The amount of wall clock time consumed by the user's applications. Many levels of aggregation are possible: for single jobs, for all the jobs, for all the jobs submitted in a given time period.
- **Job Number:** The number of grid jobs submitted by the user.
- **Physical Memory:** Physical memory used by the user's grid applications. Many levels of aggregation are possible: for single jobs, for all the jobs, for all the jobs submitted in a given time period.
- **Virtual Memory:** Virtual memory consumed by the user's grid applications. Many levels of aggregation are possible: for single jobs, for all the jobs, for all the jobs submitted in a given time period.

Grid Resource The information concerning a grid resource is similar to the information related to grid users, with the following differences:

- instead of the user name a resource account can be identified through a resource ID (such as "grid002long").
- instead of the "Certificate Subject" the "Ce ID" is used.
- instead of the assigned, booked and spent funds, the *total amount of earned grid credits* will be stored (Note: the amount earned for single jobs, that is the job cost, will be stored in the transaction information associated with the jobs).

Account Creation Accounts for User and Resources can be created via a command line interface. It is possible to manually create single accounts or, when needed, automatically create a bunch of accounts retrieving the necessary information from Virtual Organisation ldap servers (in case of user accounts) or from bdII LDAP servers (in case of resource accounts).

Creating a single user account The command used to create a single user account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAddUser [OPTIONS]
```

```

-i --interactive           Prompts for the required information.
-C --Conf <confFile>     HLR configuration file name, if different
                          from the default (/opt/glite/etc/dgas_hlr.conf).
-F --Force                Adds the record in the HLR database even if the user
                          account already exists or the corresponding
                          group or VO does not exist.
-u --uid <userID>         String that identifies the user in the database.
-e --email <address>      E-mail address of the user.
-d --descr <descr>       String that describes the user (e.g. real name).
-c --cert <certSubject>  The subject of the user's x509 certificate.
-g --gid <groupID>       Specifies the group to which the user is associated
                          in the HLR database.

```

- f --vo_id <voID> Specifies the VO to which the group is associated in the HLR database.
- a --assigned <credits> Specifies the amount of virtual credits assigned to the user.
- b --booked <credits> Specifies the amount of virtual credits already booked (shouldn't be specified manually).
- s --spent <credits> Specifies the amount of virtual credits already spent (shouldn't be specified manually).

Example: For the creation of an account for the user “guarise” belonging to the group “torino” that is part of the VO “alice”, specifying a description (i.e. the user's real name), the email address and the X509 certificate subject DN, the command line should be:

```
glite_dgas_hlrAddUser -C $GLITE_LOCATION/etc/dgas_hlr.conf \
-u guarise -g torino -f alice -d "Andrea Guarise" \
-e "andrea.guarise@to.infn.it" \
-c "/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=\
Andrea Guarise/Email=Andrea.Guarise@to.infn.it"
```

The “alice” VO ID and the “torino” group ID need to be defined creating the corresponding elements with the appropriate commands described hereafter.

Using the command with its parameter `-i` allows an interactive account creation, that is the administrator will be prompted for the required information.

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Important note: Although each account in the HLR database is associated to only one VO, the transaction information/usage records stored in the HLR database keep track of the VOMS VO for which a job was submitted even if it is different from the VO to which the account is associated on the given HLR server. See “Grid User” in Section 3.1.2.

Creating a single resource account The command used in order to create a Resource Account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAddResource [OPTIONS]
```

- i --interactive Prompts for the required information.
- F --Force Adds the record in the HLR database even if the resource account already exists or the corresponding group or VO does not exist.
- C --Conf <confFile> HLR configuration file name, if different from the default (/opt/glite/etc/dgas_hlr.conf).
- r --rid <resID> String that identifies the resource in the database.
- e --email <address> E-mail address of contact person.
- d --descr <descr> String that describes the resource.
- c --ceId <ceID> The global grid ID of the resource.
- S --certSubject <cert> The subject of the resource's host certificate.
- g --gid <groupID> Specifies the group to which the resource is associated in the HLR database.

`-f --vo_id <voID>` Specifies the VO to which the group is associated in the HLR database.

In order to create an account for a computing element with CE Id:

```
"grid002.to.infn.it:2119/jobmanager-pbs-short"
```

assigning it to the "torino" group and the "alice" Virtual Organisation, the command would be:

```
glite_dgas_hlrAddResource -r to001 -e sys.manager@to.infn.it \  
-d "grid002.to.infn.it" \  
-c "grid002.to.infn.it:2119/jobmanager-pbs-short" \  
-g torino -f alice
```

In the example the "-d" parameter specifies a description for the resource and the "-e" parameter specifies the email of the system manager. The "-r" parameter specifies an ID string that can be used to identify the resource within the HLR database.

The parameter specifying the VO is named "-f" for reasons of compatibility with its old name *fund*, in the future it might be changed into "-v". Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

The "alice" Vo ID and the "torino" group ID need to be defined creating the corresponding elements with the appropriate commands described hereafter.

Using the command with its parameter `-i` allows an interactive account creation, that is the administrator will be prompted for the required information.

Creating an account for a group of users or resources A user group can be considered a logical division of a Virtual Organization. The group "torino" belonging to the VO "alice", for example, may include all user's that collaborate with the HEP experiment ALICE and are located at Torino.

The command used to create a group of accounts is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAddGroup [OPTIONS]
```

```
-i --interactive      Prompts for the required information.  
-F --Force           Adds the record in the HLR database even if the group  
                    already exists or the corresponding VO does not exist.  
-C --Conf <confFile> HLR configuration file name, if different  
                    from the default (/opt/glite/etc/dgas_hlr.conf).  
-g --gid <groupID>  String that identifies the group.  
-d --descr <descr>  String that describes the group.  
-f --vo_id <voID>  Specifies the VO to which the group is associated  
                    in the HLR database.  
-t --total <credits> Specifies the amount of virtual credits assigned  
                    to the group.  
-b --booked <credits> Specifies the amount of virtual credits already  
                    booked (shouldn't be specified manually).  
-s --spent <credits> Specifies the amount of virtual credits already  
                    spent (shouldn't be specified manually).
```

The command for creating a group of accounts named “torino”, belonging to the Virtual Organisation “alice” would be:

```
glite_dgas_hlrAddGroup -C $GLITE_LOCATION/etc/dgas_hlr.conf \
  -g torino -d "Alice Group working in Torino/Italy" -f alice
```

In the example the “-d” options is used to specify a brief description for the group.

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”.

The “alice” Vo ID needs to be defined creating the corresponding element with the appropriate command described hereafter.

Using the command with it's parameter *-i* allows an interactive account creation, that is the administrator will be prompted for the required information.

Creating an account for a Virtual Organisation Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping. It may be considered a “supergroup” of user and/or resource accounts.

The command used to create an account for a Virtual Organisation is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAddVO [OPTIONS]
```

```

-i --interactive      Prompts for the required information.
-C --Conf <confFile> HLR configuration file name, if different
                    from the default (/opt/glite/etc/dgas_hlr.conf).
-F --Force           Adds the record in the HLR database even if the
                    VO/supergroup account already exists.
-f --vo_id <voID>   String that identifies the VO/supergroup in the database.
-d --descr <descr>  String that describes the VO/supergroup.
-t --total <credits> Specifies the total amount of virtual credits assigned
                    to the VO/supergroup.

```

Example: The command for creating the account for the above mentioned Virtual Organisation “alice” is:

```
glite_dgas_hlrAddVO -C $GLITE_LOCATION/etc/dgas_hlr.conf \
  -d "ALICE VO" -f alice -t 0
```

Where as usual the “-d” option is used to specify a description for the account, while the “-t” option specifies the initial amount of Grid Credits assigned to the VO account.

The parameter is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. This does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Using the command with it's parameter *-i* allows an interactive account creation, that is the administrator will be prompted for the required information.

Granting HLR administrator privileges An HLR administrator can remotely request information about all of the HLR's user and resource accounts and their accounted jobs.

The command used to add an HLR administrator is:

```

$GLITE_LOCATION/sbin/glite_dgas_hlrAddHlrAdmin -h

./glite_dgas_hlrAddHlrAdmin 1.0
Author: Andrea Guarise
usage:
./glite_dgas_hlrAddHlrAdmin [OPTIONS]
-i --interactive          Prompts for the admin's certificate subject.
-F --Force               Adds the record in the HLR database even
                        if the certificate subject is already listed
                        as HLR admin.
-C --Conf <confFile>    HLR configuration file name, if different
                        from the default (/opt/glite/etc/dgas_hlr.conf).
-a --admin <cert_subject> x509 certificate subject of the HLR admin.
  
```

where `-a` is used to specify the subject of the administrators personal x509 certificate. The parameter `-C` can be used to specify an HLR configuration file different from the default one. If using `-i` the command will interactively prompt for the required information. The creation of the database entry can be forced by `-F`, even if the certificate subject is already registered as an HLR administrator.

Automatic creation of a set of user accounts It is possible to automatically create a set of user accounts retrieving the necessary information from the LDAP or VOMS server of a Virtual Organisation. With this procedure, group and VO accounts as well are created when needed.

DGAS provides two different commands for the automatic import of HLR user accounts:

`glite-dgas-hlr_voImport.pl` for creating accounts for the users listed on a single VO LDAP server and `glite-dgas-hlr_userAccountImport.pl` for creating accounts for the users listed on multiple LDAP and/or VOMS servers.

glite-dgas-hlr_voImport.pl: The command for importing user accounts from a single LDAP server is:

```

> $GLITE_LOCATION/sbin/glite-dgas-hlr_voImport.pl -h
glite-dgas-hlr_voImport.pl imports HLR user accounts from an LDAP server.
Authors: A. Guarise, R. Piro.
  
```

Usage:

```

glite-dgas-hlr_voImport.pl <OPTIONS>
  
```

Where OPTIONS are:

```

-s <hostname>  Host name of the VO server to contact.
-p <port>      Port number for the VO LDAP server.
-b <ldapBase>  Base String for the LDAP query.
-v <ldapURI>   VO URI (can be used instead of -s -p -b).
-V <voID>      Virtual Organisation ID (e.g. alice, cms, atlas, lhcb...)
                used for the HLR database.
-f            Force flag, if specified existing accounts are overwritten.
  
```

Example: The command for creating the accounts for the users belonging to the Virtual Organisation of ALICE would be:

```

glite-dgas-hlr_voImport.pl \
  -v "ldap://grid-vo.nikhef.nl/o=alice,dc=eu-datagrid,dc=org" \
  -V alice

```

This commands creates the accounts for all the users registered to the ALICE Virtual Organisation. It creates the corresponding group accounts using the domain of the email address so that the users belonging to the VO are grouped by site.

glite-dgas-hlr_userAccountImport.pl: The command for importing user accounts from multiple LDAP and/or VOMS servers is:

```

> $GLITE_LOCATION/sbin/glite-dgas-hlr_userAccountImport.pl -h
glite-dgas-hlr_userAccountImport.pl imports HLR user accounts from multiple VOMS
or LDAP servers.
Authors: A. Guarise, R. Piro

```

Usage:
 glite-dgas-hlr_userAccountImport.pl [OPTIONS]

Where OPTIONS are:

```

-f          Force flag, if specified existing accounts are overwritten.
-c <file>  specifies the mkgridmap configuration file.
           Default: /opt/glite/etc/glite-dgas-hlr_mkgridmap_userAccountImport.conf
-l <file>  specifies the log file that will contain (only!) information on
           multi VO users. Normal log information is always written to stdout!
           Default: stdout

```

Since existing user accounts usually are not supposed to be overwritten (parameter `-f`), the parameter `-l` allows to specify an output file that will contain information about users that are subscribed to more than one of the VOs for which accounts are created.¹⁵ The following is an example for this “multi VO user” logfile:

```

# multi VO user logfile created by glite-dgas-hlr_userAccountImport.pl
# NOTE: users may appear more then once in this list!
# Format: |<user name>|<certificate subject>|VO(account):<vo name>|VO(other):\
<vo name>|
|Rosario Michael PIRO|/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Rosario \
Michael PIRO|VO(account):infngrid|VO(other):bio|
<...>

```

Before creating a new user account for a given VO the script checks whether the HLR database already contains an account for the same user (defined by the certificate subject) assigned to a different VO.

¹⁵It is important to note that although each account in the HLR database is associated to only one VO, the transaction information/usage records stored in the HLR database keep track of the VO for which a job was submitted even if it is different from the VO to which the account is associated on the given HLR server. See “Grid User” in Section 3.1.2.

In this case the user is considered to be a “multi VO user” and instead of creating a new account an entry is added to the multi VO user logfile, containing information on the user (user name and certificate subject), the VO to which the user is associated in the HLR database (*VO(account)*) and the VO for which the script tried to create a new account (*VO(other)*). Note that this implies that the multi VO user logfile may contain multiple entries for a user that is subscribed to more than two VOs.

Since the multiple VO user logfile should contain no entry (with distinct VOs) more than once, the file is always overwritten and never appended. Thus, if you run the script periodically (e.g. by adding it to `/etc/crontab`) you may want to make frequent backups of the multi VO user logfile (e.g. using `logrotate`).

If no logfile is specified, the information on multi VO users is written, together with the normal log information, to the standard output.

The configuration file that contains the contact strings for the LDAP and/or VOMS servers from which to import the user accounts can be specified with the parameter `-c`. In case it is not specified the default configuration file is used:

```
$GLITE_LOCATION/etc/glite-dgas-hlr_mkgridmap_userAccountImport.conf
```

Since the script uses `mkgridmap` (available from [8]) to import the HLR user accounts, the configuration file is most similar to an `mkgridmap` configuration file. It however considers only “group” entries as shown in the following example:

```
#### GROUP: group URI [lcluser]
group ldap://grid-vo.nikhef.nl/ou=lcgl,o=cms,dc=eu-datagrid,dc=org .cms
group ldap://grid-vo.cnaf.infn.it:10389/ou=Testbed-Firb,o=gridit,c=it .gridit
group vomss://kuiken.nikhef.nl:8443/voms/EGEE .EGEE
```

This example shows the contact strings required for the different protocols: If querying an LDAP server (first two entries of the example) the LDAP search base has to be specified. If the LDAP server doesn't use the standard port (389), the port has to be specified as in the contact string of the second entry. VOMS servers can be queried by invoking `/services/VOMSCompatibility?method=getGridmapUsers` (third entry in the example, the translation to the method call will be done by `glite-mkgridmap`).

In all cases the name of the VO is specified after the contact string and is always indicated by a dot (that is “`.<vo name>`” instead of “`<vo name>`”).

Automatic creation of a set of resource accounts It is also possible to automatically create a set of resource accounts using a `bdII` LDAP server as the source of information. The command is:

```
> $GLITE_LOCATION/sbin/glite-dgas-hlr_bdiiImport.pl -h
glite-dgas-hlr_bdiiImport.pl imports HLR resource accounts from a BDII server.
Usage:
glite-dgas-hlr_bdiiImport.pl <OPTIONS>
```

Where `OPTIONS` are:

```
-s <hostname>  Host name of the BDII server to contact.
-p <port>      Port number for the BDII server to query.
-b <base>      Base string for the query (Default="o=Grid").
-v <voFlag>    Virtual Organisation flag ("bdII" or "Mds-Vo-Name").
               "bdII": Selects the bdII FQDN as the resource's VO ID.
```

"Mds-Vo-Name": Selects the Mds-Vo-Name record
 as the VO ID to which the resource is
 associated in the HLR database.

Default: "Mds-Vo-Name".

-f Force flag, if specified existing accounts are overwritten.

The command to create the resource accounts for the resources registered in the 'GILDA' bdII (without overwriting already existing accounts) would be:

```
> $GLITE_LOCATION/sbin/glite-dgas-hlr_bdiiImport.pl \  

  -s grid017.ct.infn.it -p 2170 -b "o=Grid"
```

This command creates the accounts for the registered resources. It also creates the VO and group accounts used to divide the resources into subsets. The concept of VO is not always applicable to a resource set, but it can be used as a sort of meta-group. By default the command creates VO accounts using the Mds-Vo-Name parameter of the resource entry in the bdII. It is also possible to create an unique VO account for all the resources registered in the bdII, this is done specifying the `-v bdII` parameter.

The group accounts are created using the domain address of the resource host name.

Account Deletion The commands for the deletion of HLR accounts are similar to those described in Section 3.1.2, this section therefore gives only brief descriptions. All commands, however, except the `-h` option, that displays the available command line arguments, and the possibility to interactively prompt the user for the required information (`-i` option).

Deleting a user account The command used to delete a single user account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrDelUser
```

```
glite_dgas_hlrDelUser 1.0
Authors: Stefano Barale, Andrea Guarise
usage:
glite_dgas_hlrDelUser [OPTIONS]
-i --interactive           Prompts for the required information.
-C --Conf <confFile>     HLR configuration file name, if different
                          from the default (/opt/glite/etc/dgas_hlr.conf).
-u --uid <userID>        String that identifies the user in the HLR
                          database (mandatory).
-e --email <address>     E-mail address of the user.
-d --descr <descr>       String that describes the user (e.g. real name).
-c --cert <certSubject>  The subject of the user's x509 certificate.
-g --gid <groupID>       Specifies the group to which the user is associated
                          in the HLR database (mandatory).
-f --vo_id <voID>       Specifies the VO to which the group is associated
                          in the HLR database (mandatory).
```

Note that several options are mandatory in order to avoid the unintentional deletion of user accounts (in case a user belongs to several groups and/or VOs the specification of a single parameter, such as the uid, would require more attention, while additionally specifying the group and VO IDs helps to avoid errors).
 Example:

```

glite_dgas_hlrDelUser -C $GLITE_LOCATION/etc/dgas_hlr.conf \
    -u guarise -g torino -f alice -d "Andrea Guarise" \
    -e "andrea.guarise@to.infn.it" \
    -c "/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=\
Andrea Guarise/Email=Andrea.Guarise@to.infn.it"
  
```

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Deleting a resource account The command used to delete a single resource account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrDelResource
```

```

glite_dgas_hlrDelResource
Version: 1.0
Authors: Stefano Barale and Andrea Guarise
usage:
glite_dgas_hlrDelResource [OPTIONS]
-i --interactive           Prompts for the required information.
-C --Conf <confFile>     HLR configuration file name, if different
                           from the default (/opt/glite/etc/dgas_hlr.conf).
-r --rid <resID>         String that identifies the resource in the HLR
                           database (mandatory).
-d --descr <descr>      String that describes the resource.
-c --ceId <ceID>        The global grid ID of the resource.
-g --gid <groupID>      Specifies the group to which the resource is
                           associated in the HLR database (mandatory).
-f --vo_id <voID>       Specifies the VO to which the group is associated
                           in the HLR database (mandatory).
  
```

Note that several options are mandatory in order to avoid the unintentional deletion of resource accounts.
Example:

```

glite_dgas_hlrDelResource -r to001 -d "grid002.to.infn.it" \
    -c "grid002.to.infn.it:2119/jobmanager-pbs-short" \
    -g torino -f alice
  
```

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Deleting an account for a group of users or resources The command used to delete a group account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrDelGroup -h
```

```
glite_dgas_hlrDelGroup 1.0
```

Authors: Stefano Barale and Andrea Guarise

usage:

```

glite_dgas_hlrDelGroup [OPTIONS]
-i --interactive      Prompts for the required information.
-C --Conf <confFile> HLR configuration file name, if different
                    from the default (/opt/glite/etc/dgas_hlr.conf).
-g --gid <groupID>   String that identifies the group (mandatory).
-f --vo_id <voID>    Specifies the VO to which the group is associated
                    in the HLR database (mandatory).
-d --descr <descr>   String that describes the group.
  
```

Note that several options are mandatory in order to avoid the unintentional deletion of group accounts.

Example:

```

glite_dgas_hlrDelGroup -C $GLITE_LOCATION/etc/dgas_hlr.conf \
  -g torino -d "Alice Group working in Torino/Italy" -f alice
  
```

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Deleting an account for a Virtual Organization The command used to delete a VO account is:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrDelVO
```

```

glite_dgas_hlrDelVO 1.0
Author: Andrea Guarise, Stefano Barale
usage:
glite_dgas_hlrDelVO [OPTIONS]
-i --interactive      Prompts for the required information.
-C --Conf <confFile> HLR configuration file name, if different
                    from the default (/opt/glite/etc/dgas_hlr.conf).
-f --vo_id <voID>    String that identifies the VO/supergroup in the HLR
                    database (mandatory).
-d --descr <descr>   String that describes the VO/supergroup.
  
```

Example:

```

glite_dgas_hlrDelVO -C $GLITE_LOCATION/etc/dgas_hlr.conf \
  -d "ALICE VO" -f alice
  
```

The parameter specifying the VO is named “-f” for reasons of compatibility with its old name *fund*, in the future it might be changed into “-v”. Note that this does not correspond to the VOMS VO, but is an HLR database specific classification for the purpose of account grouping.

Removing HLR administrator privileges An HLR administrator can remotely request information about all of the HLR's user and resource accounts and their accounted jobs.

The command used to delete an HLR administrator is:

```

$GLITE_LOCATION/sbin/DelHlrAdmin -h

./glite_dgas_hlrDelHlrAdmin 1.0
Authors: Andrea Guarise
usage:
./glite_dgas_hlrDelHlrAdmin [OPTIONS]
-i --interactive           Prompts for the admin's certificate subject.
-C --Conf <confFile>     HLR configuration file name, if different
                           from the default (/opt/glite/etc/dgas_hlr.conf).
-a --admin <cert_subject> x509 certificate subject of the HLR admin.
  
```

where `-a` is used to specify the subject of the administrators personal x509 certificate. The parameter `-C` can be used to specify an HLR configuration file different from the default one. If using `-i` the command will interactively prompt for the required information.

3.1.3 ADMINISTRATION TOOLS FOR DATABASE QUERIES

This Section describes how an HLR admin can query the HLR server for the stored information. DGAS offers different commands (local on the HLR server¹⁶) to retrieve the information.

Account Monitoring Accounts can be monitored by a local user on the HLR server using the following command line tools:

glite_dgas_hlrQueryUser This command allows to query user accounts on the local HLR server. It can be used to retrieve information for single user accounts as well as to list all user accounts.

The command takes the following command line arguments:

```

> $GLITE_LOCATION/sbin/glite_dgas_hlrQueryUser -h
glite_dgas_hlrQueryUser
Version: 1.0
Author: Andrea Guarise
28/08/2001

usage:
glite_dgas_hlrQueryUser [OPTIONS]
-i --interactive           Prompts for the required information.
-C --Conf <confFile>     HLR configuration file name, if different
                           from the default (/opt/glite/etc/dgas_hlr.conf).
-T --transactions        Display the list of transactions (jobs).
-U --users                Display user account information.
-a --all                  Display information for all users.
-u --uid <userID>        String that identifies the user in the database.
-e --email <address>     E-mail address of the user.
-c --cert <certSubject> The subject of the user's x509 certificate.
  
```

¹⁶To retrieve the information from remote hosts use the HLR clients described in Section 3.1.1.

The parameter `-T` can be used to include transaction (job) information for the selected user account. The parameter `-U` instead retrieves information about the user itself. At least one of both parameters should be specified, otherwise the command will only check the existence of the specified account (exit status 0 if present in the HLR database).

A particular user can be specified by either using parameter `-c` (the certificate subject of the user), parameter `-e` (the user's e-mail address, if present in the database) or parameter `-u` (the ID of the user's account in the HLR database). To retrieve information for all users, the parameter `-a` is used.

The parameter `-C`, finally, can be used to specify a HLR configuration file different from the default one.

The following examples demonstrate the most frequent use cases:

a) Getting a brief summary of the account status of a single user:

```
> lite_dgas_hlrQueryUser -U -e "patania@to.infn.it"
|GiuseppePatania|patania@to.infn.it|Giuseppe Patania|/C=IT/O=INFN/OU=\
Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=patania@to.infn.it|\
to.infn.it|EGEE|0|0|0|wall_time=41,cpu_time=7,job_number=7|
```

In this example the command returns a line similar to the output of the remote `glite_dgas_hlrUserInfoClient` described in Section 3.1.1. However, it additionally includes information on the VO for which the user account is registered (in this case EGEE).

b) Getting a list of all user accounts with a brief summary of the account status:

```
> glite_dgas_hlrQueryUser -Ua
|AndreaGuarise|Andrea.Guarise@to.infn.it|Andrea Guarise|/C=IT/O=INFN/OU=\
Personal Certificate/L=Torino/CN=Andrea Guarise/Email=Andrea.Guarise@to.infn.it|\
to.infn.it|EGEE|0|0|0|wall_time=97380,cpu_time=48173,job_number=3319|
|GiuseppePatania|patania@to.infn.it|Giuseppe Patania|/C=IT/O=INFN/OU=\
Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=patania@to.infn.it|\
to.infn.it|EGEE|0|0|0|wall_time=41,cpu_time=7,job_number=7|
<...>
```

In this example the command returns one line for each user account, similar to the output of the remote `glite_dgas_hlrUserInfoClient` described in Section 3.1.1. However, it additionally includes information on the VO for which the user account is registered (in this case EGEE).

c) Getting a list of the transactions (jobs) associated with a single user account:

```
> glite_dgas_hlrQueryUser -T -e "patania@to.infn.it"
|36493|out|https://gundam.cnaf.infn.it:9000/k44VpYeOY9Ua6HPUZk-SRg|\
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=\
patania@to.infn.it||lxb2077.cern.ch:2119/blah-pbs-long|\
hlr02.to.infn.it:56568:|0|1120475773|
|36494|out|https://gundam.cnaf.infn.it:9000/w1jRqIkq6lY3t_83fHDHTw|\
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=\
patania@to.infn.it||lxb2077.cern.ch:2119/blah-pbs-short|\
hlr02.to.infn.it:56568:|0|1120475806|
<...>
```

In this example the command returns one line for each transaction/job.¹⁷ The single fields are separated by pipes (|) and contain the following information:

¹⁷Note: There is no possibility to filter the output, thus for retrieving more specific transactions please use the `glite_dgas_hlrAdvancedQuery` described below.

- the transaction ID in the HLR database (here: 36493)
- the transaction type (always out=debit for users and in=credit for resources)
- the global job ID (here: https:// ...)
- the certificate subject of the debited user (here: /C=IT/O=INFN/OU=Personal ...)
- the HLR contact string of the User HLR (in case of transactions of type in)
- the grid ID of the credited resource (here: lxb2077.cern.ch ...)
- the HLR contact string of the Resource HLR (in case of transactions of type out; here: hlr02.to.infn.it:56568:)
- the job cost in grid credits (here: 0)
- the time stamp of the transaction (here: 1120475773)

glite_dgas_hlrQueryResource This command allows to query resource accounts on the local HLR server. It can be used to retrieve information for single resource accounts as well as to list all resource accounts.

The command takes the following command line arguments:

```
> $GLITE_LOCATION/sbin/glite_dgas_hlrQueryResource -h

glite_dgas_hlrQueryResource 1.0
Author: Andrea Guarise
usage:
./glite_dgas_hlrQueryResource [OPTIONS]
-i --interactive           Prompts for the required information.
-C --Conf <confFile>     HLR configuration file name, if different
                           from the default (/opt/glite/etc/dgas_hlr.conf).
-T --transactions        Display the list of transactions (jobs).
-R --resources            Display resource account information.
-a --all                  Display information for all resources.
-r --rid <resID>         String that identifies the resource in the database.
-c --ceId <ceID>        The global grid ID of the resource.
```

The parameter `-T` can be used to include transaction (job) information for the selected resource account. The parameter `-R` instead retrieves information about the resource itself. At least one of both parameters should be specified, otherwise the command will only check the existence of the specified account (exit status 0 if present in the HLR database).

A particular resource can be specified by either using parameter `-c` (the resource's global CE ID) or parameter `-r` (the ID of the resource's account in the HLR database). To retrieve information for all resources, the parameter `-a` is used.

The parameter `-C`, finally, can be used to specify a HLR configuration file different from the default one.

The following examples demonstrate the most frequent use cases:

a) Getting a brief summary of the account status of a single resource:

```
> glite_dgas_hlrQueryResource -R -c "lxb2077.cern.ch:2119/blah-pbs-short"
|lxb2077C|alessio.gianelle@pd.infn.it|lxb2077.cern.ch:2119/blah-pbs-short|\
lxb2077.cern.ch:2119/blah-pbs-short|proto|egee|0|wall_time=465890,cpu_time=\
46251,job_number=14061|
```

The single fields of the returned output are separated by pipes (|) and contain the following information:

- resource id in the HLR database (here: lxb2077C)
- e-mail address of the resource administrator (here: alessio.gianelle@pd.infn.it)
- description of resource (here: lxb2077.cern.ch:2119/blah-pbs-short)
- global CE ID (here: lxb2077.cern.ch:2119/blah-pbs-short)
- group in the HLR database (here: proto)
- VO to which the resource is associated in the HLR database (here: egee)
- total grid credits earned (here: 0)
- total wall clock time, CPU time and number of executed jobs
(here: wall_time=465890,cpu_time=46251,job_number=14061)

b) Getting a list of all resource accounts with a brief summary of the account status:

```
> glite_dgas_hlrQueryResource -Ra
|lxde01E|alessio.gianelle@pd.infn.it|lxde01.pd.infn.it:2119/blah-pbs-pbsque|\
lxde01.pd.infn.it:2119/blah-pbs-pbsque|proto|egee|0|wall_time=45849,cpu_time=\
12281,job_number=1430|
|lxb2022A|alessio.gianelle@pd.infn.it|lxb2022.cern.ch:2119/blah-lsf-jra1_high|\
lxb2022.cern.ch:2119/blah-lsf-jra1_high|proto|egee|0|wall_time=1692,cpu_time=\
68,job_number=117|
<...>
```

In this example the command returns one line for each resource account. Each line contains the fields described in the previous example.

c) Getting a list of the transactions (jobs) associated with a single resource account:

```
> glite_dgas_hlrQueryResource -T -c "lxb2022.cern.ch:2119/blah-lsf-jra1_low"
Last Transactions:
|37214|in|https://gundam.cnaf.infn.it:9000/A64bCdxLiWwOM-_3h9B52g|/C=IT/O=\
INFN/OU=Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=\
patania@to.infn.it|local|lxb2022.cern.ch:2119/blah-lsf-jra1_low||0|1118778504|
|38310|in|https://gundam.cnaf.infn.it:9000/hSe7VxlaSYiYG9qFvzHreQ/1|/C=IT/O=\
INFN/OU=Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=\
patania@to.infn.it|local|lxb2022.cern.ch:2119/blah-lsf-jra1_low||0|1119046168|
<...>
```

In this example the command returns one line for each transaction/job.¹⁸ The single fields are separated by pipes (|) and contain the following information:

¹⁸Note: There is no possibility to filter the output, thus for retrieving more specific transactions please use the `glite_dgas_hlrAdvancedQuery` described below.

- the transaction ID in the HLR database (here: 37214)
- the transaction type (always out=debit for users and in=credit for resources)
- the global job ID (here: https:// ...)
- the certificate subject of the debited user (here: /C=IT/O=INFN/OU=Personal ...)
- the HLR contact string of the User HLR (in case of transactions of type in; here: local, indicating that the user's account is on the same HLR)
- the grid ID of the credited resource (here: lxb2022.cern.ch ...)
- the HLR contact string of the Resource HLR (in case of transactions of type out)
- the job cost in grid credits (here: 0)
- the time stamp of the transaction (here: 1118778504)

glite_dgas_hlrQueryGroup This command allows to query for groups of user and resource accounts on the local HLR server. It can be used to retrieve information for single groups as well as to list all groups.

The command takes the following command line arguments:

```
> $GLITE_LOCATION/sbin/glite_dgas_hlrQueryGroup -h

glite_dgas_hlrQueryGroup 1.0
Author: Andrea Guarise
usage:
./glite_dgas_hlrQueryGroup [OPTIONS]
-i --interactive      Prompts for the required information.
-C --Conf <confFile> HLR configuration file name, if different
                    from the default (/opt/glite/etc/dgas_hlr.conf).
-a --all             Display information for all groups.
-g --gid <groupID>  String that identifies the group.
```

A particular group can be specified by using the parameter `-g` (the ID of the group in the HLR database). To retrieve information for all groups, the parameter `-a` is used.

The parameter `-C`, finally, can be used to specify a HLR configuration file different from the default one.

The following examples demonstrate the most frequent use cases:

a) Getting a brief summary of the status of a single group:

```
> glite_dgas_hlrQueryGroup -g torino
|torino|JRA1 Group working in Torino/Italy|jra1|1000|
```

The single fields of the returned output are separated by pipes (|) and contain the following information:

- group ID in the HLR database (here: torino)
- description of the group (here: JRA1 Group ...)
- VO to which the group is associated in the HLR database (here: jra1)

- total grid credits assigned (here: 1000)

b) *Getting a list of all groups:*

```
> glite_dgas_hlrQueryGroup -a
|torino|JRA1 Group working in Torino/Italy|jra1|1000|
|to.infn.it|EGEE users at INFN Turin|EGEE|5000|
|egtest||egtest|0|
<...>
```

In this example the command returns one line for each group. Each line contains the fields described in the previous example.

glite_dgas_hlrQueryVO This command allows to query for VOs (that can be considered super-groups of user and resource accounts) on the local HLR server. It can be used to retrieve information for single VOs as well as to list all VOs.

The command takes the following command line arguments:

```
> $GLITE_LOCATION/sbin/glite_dgas_hlrQueryVO -h

glite_dgas_hlrQueryVO 1.0
Author: Andrea Guarise
usage:
glite_dgas_hlrQueryVO [OPTIONS]
-i --interactive      Prompts for the required information.
-C --Conf <confFile> HLR configuration file name, if different
                    from the default (/opt/glite/etc/dgas_hlr.conf).
-a --all             Display information for all VOs/supergroups.
-f --vo_id <voID>   String that identifies the VO/supergroup in the database.
```

A particular VO can be specified by using the parameter `-f` (the ID of the VO in the HLR database). To retrieve information for all VOs, the parameter `-a` is used.

Note: The parameter specifying the VO is named “`-f`” for reasons of compatibility with its old name *fund*, in the future it might be changed into “`-v`”.

The parameter `-C`, finally, can be used to specify a HLR configuration file different from the default one.

The following examples demonstrate the most frequent use cases:

a) *Getting a brief summary of the status of a single VO:*

```
> glite_dgas_hlrQueryVO -f jra1
|jra1|JRA1 developers|10000|
```

The single fields of the returned output are separated by pipes (`|`) and contain the following information:

- VO ID in the HLR database (here: `jra1`)
- description of the VO (here: `JRA1 developers`)
- total grid credits assigned (here: `10000`)

b) Getting a list of all VOs:

```

> glite_dgas_hlrQueryVO -a
|EGEE|VO for EGEE users|50000|
|jral|JRA1 developers|10000|
<...>
  
```

In this example the command returns one line for each VO. Each line contains the fields described in the previous example.

glite_dgas_hlrQueryHlrAdmin This command allows to query for registered HLR administrators (that can access all accounting information) on the local HLR server. It can be used to verify the registration of a single administrator as well as to list all registered administrators.

The command takes the following command line arguments:

```

> $GLITE_LOCATION/sbin/glite_dgas_hlrQueryHlrAdmin -h

glite_dgas_hlrQueryHlrAdmin 1.0
Author: Andrea Guarise
usage:
glite_dgas_hlrQueryHlrAdmin [OPTIONS]
-C --Conf <confFile>      HLR configuration file name, if different
                           from the default (/opt/glite/etc/dgas_hlr.conf).
-a --all                  Display the list of all HLR admins.
-A --admin <cert_subject> x509 certificate subject of the HLR admin.
  
```

A particular administrator can be specified by using the parameter `-A` (the administrator's certificate subject). To retrieve a list of all administrators, the parameter `-a` is used.

The parameter `-C`, finally, can be used to specify a HLR configuration file different from the default one.

The following examples demonstrate the most frequent use cases:

a) Verifying the registration of a single HLR administrator:

```

> glite_dgas_hlrQueryHlrAdmin -A "/C=IT/O=INFN/OU=Personal Certificate/L=\
Torino/CN=Rosario Michael PIRO"
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Rosario Michael PIRO
  
```

In case the certificate subject is registered as an HLR administrator, the command returns the certificate subject itself, otherwise it gives an error.

b) Getting a list of all HLR administrators:

```

> glite_dgas_hlrQueryHlrAdmin -a
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Andrea Guarise/Email=\
Andrea.Guarise@to.infn.it
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Giuseppe Patania/Email=\
patania@to.infn.it
/C=IT/O=INFN/OU=Personal Certificate/L=Torino/CN=Rosario Michael PIRO
  
```

In this example the command returns one line for each HLR administrator. Each line contains the certificate subject of the administrator.

glite_dgas_hlrAdvancedQuery This command line tool can be used for more generic and complex queries to retrieve lists of transactions as well as aggregated information on users, resources, groups and VOs.

The program can be usually found locally on the HLR server at:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrAdvancedQuery
```

and has the following options:

```
> $GLITE_LOCATION/sbin/glite_dgas_hlrAdvancedQuery -h
```

```
./glite_dgas_hlrAdvancedQuery
```

```
Version: 1.0PROTOTYPE
```

```
Author: Andrea Guarise
```

```
13/09/2004
```

```
usage:
```

```
./glite_dgas_hlrAdvancedQuery [OPTIONS]
```

```
-Q --queryType <queryType>    Specifies the type of query
```

```
-e --email <emailAddress>      Specifies the email address of the user
```

```
-u --userCert <cert_subject>   Used to specify the user's certificate subject
```

```
-r --resourceId <ceID>         Used to specify the global CE ID for the resource
```

```
-v --voId <voID>               Used to specify the virtual organization
```

```
-g --groupId <groupID>         Used to specify the ID for the group
                                (Note: -g refers to the group account in the HLR
                                database, not the VOMS group!)
```

```
-t --time <interval>           Time interval to use in the query, in the form:
                                "<startTimeStamp>-<endTimeStamp>" : all the info in the given period
                                "-<endTimeStamp>" : all the info available until endTimeStamp
                                "<startTimeStamp>-" : all the info available from startTimeStamp
                                Timestamps are specified in seconds since Jan. 1, 1970 0:00:00 GMT
                                If not specified or '*': no filtering for the time period.
```

```
-F --Frequency <freq>         Frequency for the aggregation of query results
```

```
-D --debug                      Ask for debug output
```

```
-E --Extended                   Show more info
```

```
-C --Conf                       HLR configuration file, if not the default: /opt/glite/etc/dgas_
```

```
-h --help                       This help
```

The parameter <queryType> can be one of the following:

```
"userAggregate":               Aggregate usage records for a user
```

```
"resourceAggregate":           Aggregate usage records for a resource
```

```
"groupAggregate":              Aggregate usage records for a group of accounts
```

```
"voAggregate":                 Aggregate usage records for the users belonging to a VO
```

```
"userJobList":                 List of jobs submitted by a user
```

```
"resourceJobList":             List of jobs executed by a resource
```

The parameter <freq> can be one of the following:

```
"day":                          daily aggregate usage records
```

"week": weekly aggregate usage records
 "month": monthly aggregate usage records
 "hour": hourly aggregate usage records

The command line tool is equivalent to the remote client `glite_dgas_hlrAdvancedQuery` (described in Section 3.1.1, see there). The option `--jobId`, however, has not yet been implemented.

3.1.4 ADMINISTRATION TOOLS FOR DATABASE CONVERSIONS

Although DGAS HLR servers accept Usage Records in the format specified by the GGF Usage Record Work Group (UR-WG), that is based on XML documents, it stores the data in a different format in a relational database.

To improve interoperability with other systems conversion tools for the accounting database are currently being developed.

glite-dgas-convertDGAS2APEL.pl This is a tool that allows to convert the DGAS accounting records to the `LcgRecords` used by APEL [11]. Run on an HLR server it converts all transactions related to the HLR's resource accounts. Transactions related to user accounts are not converted, since each transaction is registered twice, once for the user that submitted a job (on the User HLR) and once for the resource that executed it (on the Resource HLR).

The script can usually be found locally on the HLR server (although it may also be run on a remote host if it can access the HLR's database¹⁹) at:

```
$GLITE_LOCATION/sbin/glite-dgas-convertDGAS2APEL.pl
```

and has the following options:

```
> $GLITE_LOCATION/sbin/glite-dgas-convertDGAS2APEL.pl -h
Author: Rosario Piro (piro@to.infn.it)
```

```
Usage:
glite-dgas-convertDGAS2APEL.pl [OPTIONS]
```

Where OPTIONS are:

```

-c <file>    Specifies the configuration file.
              Default: $GLITE_LOCATION/etc/glite-dgas-convertDGAS2APEL.conf
-s <tid>     HLR transaction id from which to start the conversion
              Default: Determined from the last entry in the ConversionLog
              table: start point = highest tid (converted) + 1
-e <tid>     HLR transaction id up to which to convert
              Default: Determined from the HLR database:
              end point = highest tid found in the trans_in table
-n           No detailed log information in table MAP_tid_RecordIdentity
              (summarized log information in ConversionLog will always
              be written).
-T <ceType>  Type of CEs for which accounting information is to be converted.
              Possible values for <ceType>:
```

¹⁹We however prefer not to let the HLR server's MySQL listen for remote connections.

- "edg": a GRIS can be contacted for each CE to retrieve some information on the CE's characteristics (e.g. SiteName; in case they are not provided by the accounting record itself).
- Default: none, only accounting information contained in the HLR database will be used.
- M Keep a mapping from CE IDs to ExecutingSites up-to-date and use that mapping in case the site name cannot be retrieved in any other way (e.g. querying the GRIS for EDG resources).
- R Retry to construct and insert LcgRecords that could not be successfully converted and inserted into the APEL database in previous executions of glite-dgas-convertDGAS2APEL.pl (up to MAX_NUM_RETRIES). If a construction/insertion attempt is not successful, it will be tracked in ConversionErrors only if -R is specified.
- v Verbose output: additional information of conversion of the single LcgRecord fields will be printed to stdout.

The option `-c` can be used to specify a configuration file different from the default.

The parameters `-s` and `-e` allow to specify the interval of transaction IDs to be translated (each transaction is assigned an internal transaction ID, or `tid`, by the HLR server). If the start `tid` is omitted the tool will try to determine the `tid` of the last translated transaction (see below) and will start with the next transaction. If the end `tid` is omitted the tool will convert up to the last transaction in the HLR's database.

The parameter `-n` allows to skip the logging of the detailed mapping of HLR transaction IDs to APEL LcgRecord IDs (see below).

Additionally to the logging and ID mapping information stored in the database, a log of the conversion process will be written to the standard output and may be redirected to a file if required.

Using parameter `-v` will add additional information about the conversion of the single LcgRecords fields to the standard output. Note: The normal log information is abundant and in most cases sufficient, `-v` can be interesting for debugging purpose.

Some additional information for the single transactions that have to be converted into the APEL LcgRecords format may be obtained from external sources that depend on the CEs' type for which data is stored in the HLR database. For resources that have a GRIS on the default port 2135 (EDG resources) the argument `"-T edg"` may be specified on the command line. This will cause the conversion tool to contact the GRIS to retrieve the name of the site (`GlueSiteName`, `siteName` or `GlueSiteUniqueID`) that is required for the `ExecutingSite` field of the LcgRecords entry.

If `-M` is specified on the command line, the tool will keep an up-to-date mapping from CE IDs to ExecutingSites (site names, see above) in the conversion log database. This mapping will be used in case the site name cannot be determined from the transaction log or external sources (e.g. the GRIS of an EDG resource may be temporarily unavailable).

Specifying the parameter `-R` will let the conversion tool keep track of error conditions in the conversion log database. Error conditions can occur when APEL records cannot be constructed due to missing information²⁰ (such as the `ExecutingSite`, see above) or cannot be inserted into the APEL database, e.g. due to failed connections. In these cases an error entry for the specific transaction will be stored in the conversion log database. Furthermore, specifying `-R` on the command line will cause the conversion tool to retry the conversion of transactions with error conditions from previous runs before processing the

²⁰Please note that only essential LcgRecords fields are considered mandatory.

transactions specified by `-s` and `-e`. If the maximum number of retries (see below) has been reached, the transactions will be inserted (if possible) into the APEL database even if they are incomplete (e.g. missing SiteName).

The configuration file for the conversion tool looks like this:

```

#
# Configuration file for conversion of DGAS schema to APEL schema
#

#
# Database for converted accounting information.
# This is the database where the APEL LcgRecords are written.
# ATTENTION: Do NOT use the original APEL database here, otherwise
# drop_db_dgas2apel_APEL-DB.sh will drop the entire APEL database!!!
# ALTERNATIVELY use a user without the permission to alter the DB, if
# the information should be written directly to the original APEL database.
#
APEL_DB_SERVER="localhost"
APEL_DB_PORT="3306"
APEL_DB_NAME="dgas2apel"
APEL_DB_USER="root"
APEL_DB_PASSWORD=""

#
# Conversion log database (usually equal to the APEL conversion db if local)
# This is the database where the conversion log is written to keep track of
# already converted HLR transactions.
# ATTENTION: Do NOT use the HLR database for this, otherwise
# drop_db_dgas2apel_ConversionLog-DB.sh will drop the entire HLR database!!!
# ALTERNATIVELY use a user without the permission to alter the DB, if
# the log information should be written directly to the original HLR database.
#
CONV_DB_SERVER="localhost"
CONV_DB_PORT="3306"
# Do NOT use "hlr" for CONV_DB_NAME if the user has the permission to drop it!
CONV_DB_NAME="dgas2apellog"
CONV_DB_USER="root"
CONV_DB_PASSWORD=""

#
# DGAS HLR database (source of accounting information)
# This is the database where the original HLR accounting information is taken
# from.
#
HLR_DB_SERVER="localhost"
HLR_DB_PORT="3306"
HLR_DB_NAME="hlr"
HLR_DB_USER="root"
HLR_DB_PASSWORD=""

#
# Configuration of the conversion procedure itself:
#
# Number of retries per transaction, before giving up (one retry per execution
# of the conversion tool); integer value specified as a string (default: 10)
MAX_NUM_RETRIES="10"

#
# Additional info required for the LcgRecords table
#

```

```

# This will be used to construct a unique RecordIdentity:
# You have to specify the HLRs contact string here:
#HLR_LOCATION="<HLR_hostname>:<port>:<subject_of_host_certificate>"
  
```

The HLR_LOCATION has to be the contact string of the HLR server from which the accounting information is taken. This information will be used to build a unique RecordIdentity for the APEL LcgRecords entry.

The maximum number of retries before forcing the insertion of incomplete LcgRecords entries into the APEL database (see above), is defined by MAX_NUM_RETRIES.

As can be seen in the configuration file the script requires access to three databases:

- The HLR database from which the accounting information is taken (access via SELECT is enough).
- The APEL database in which the generated LcgRecords are stored (access via INSERT is enough).
- A third database (CONV) that is used to store logging information for the conversion process (access via SELECT, INSERT and in case of MAP_CEID_ExecutingSite also DELETE). This database contains the following tables:
 - ConversionLog, that keeps a record for each past conversation process, storing above all the interval of tids of the translated transactions (such that the conversion process can begin from the first transaction not yet translated, if the parameter `-s` is omitted, see above).
 - ConversionErrors, that keeps track of HLR transactions that could not be successfully converted into LcgRecords entries and/or that could not be successfully inserted into the APEL database (used only if `-R` is specified on the command line, see above).
 - MAP_tid_RecordIdentity, a table that keeps a detailed mapping of HLR_LOCATIONS and HLR transaction IDs to RecordIdentities of the generated APEL LcgRecords, if the parameter `-n` is not omitted (see above).
 - MAP_CEID_ExecutingSite, that is used to keep an up-to-date mapping from CE IDs to site names in case they are not specified in the transaction log and cannot be retrieved from external resources, such as a GRIS for EDG resources (used only if `-M` is specified on the command line, see above).

Additionally to the conversation tool, four scripts for the creation/deletion of the APEL and CONV databases are provided:

- `glite-dgas-create_db_dgas2apel_APEL-DB.sh`: creates the APEL database for the LcgRecords entries.
- `glite-dgas-create_db_dgas2apel_ConversionLog-DB.sh`: creates the CONV database for the conversion log information.
- `glite-dgas-drop_db_dgas2apel_APEL-DB.sh`: deletes the APEL database.
- `glite-dgas-drop_db_dgas2apel_ConversionLog-DB.sh` deletes the CONV database.

These scripts take no input parameters, but instead read the configuration file described above (Be sure to edit the configuration file BEFORE using them).

3.1.5 ADMINISTRATION TOOLS FOR DATABASE UPDATES

The structure of the database underlying the HLR server may be (and has been) improved and thus slightly change. We thus provide a tool that, when launched checks whether the database structure is compatible with the present HLR version and eventually applies the necessary changes:

glite-dgas-hlr_updateDB.pl The script that automatically updates the MySQL database structure if necessary is:

```
$GLITE_LOCATION/sbin/glite-dgas-hlr_updateDB.pl
```

and has the following options:

```
> $GLITE_LOCATION/sbin/glite-dgas-hlr_updateDB.pl -h
Author: Rosario Piro (piro@to.infn.it)
```

Usage:
glite-dgas-hlr_updateDB.pl [OPTIONS]

Where OPTIONS are:

```
-c <file>    Specifies the configuration file.
              Default: $GLITE_LOCATION/etc/dgas_hlr.conf
-r <passwd>  Specifies the MySQL root password. If not specified on the
              command line, the tool tries to use the HLR MySQL user/password
              specified in the HLR configuration file. This will work only, if
              HLR MySQL user has the privileges to modify the HLR databases.
-v          Verbose output: more detailed information written to stdout.
```

The parameter `-c` may be used to specify an HLR configuration file (containing the information on the underlying database) other than the default configuration file. If the MySQL root user shall be used to do the eventually necessary changes then the password can be specified with `-r` otherwise the tool attempts to use the user/password information contained in the HLR configuration file (which requires the specified user to have the privilege to modify the database).

Note that for each possible change the tool will first check whether it is necessary or whether the database is up-to-date with respect to the specific change. This means that the tool may be used to update older databases from different HLR versions (including old EDG versions).

3.2 APPLICATION PROGRAM INTERFACES

Additionally to the CLI tools, the DGAS HLR furnishes a C++ API that can be used to contact an HLR server. In this Section we briefly describe the C++ API for the most common service requests.

3.2.1 PINGING THE HLR SERVER

The header file of the client that is used to ping the HLR server and retrieve its status is

```
interface/glite/dgas/hlr-clients/ping/pingClient.h
```

It defines structures that contain the information regarding the number of requests to the HLR server, as well as the number of errors that occurred (e.g. due to expired proxies). The HLR pingEngine is briefly described in Section 1.3.1. Furthermore the header defines a function that is invoked to retrieve the server status:

```

struct statusInfo {
    string engines;
    int uiRequests;
    int ATMRequests;
    int bankRequests;
    int jobAuthRequests;
    int userAuthRequests;
    int paRequests;
    int pingRequests;
};

struct errorInfo {
    string engines;
    int uiErrors;
    int ATMErrors;
    int bankErrors;
    int jobAuthErrors;
    int userAuthErrors;
    int paErrors;
    int pingErrors;
};

int dgas_ping_client(string &acct_id,
                    int pingType,
                    statusInfo *status,
                    errorInfo *errors,
                    string *server_answer);

```

The `acct_id` is the contact string (see Section 3.1.1 for how to form a contact string) of the HLR server. The `pingType` can be 0 (no detailed status info requested) or 1 (detailed status info requested). If detailed status information is requested the two structures `status` and `errors` will be filled with the desired information, that is the number of requests and errors per HLR engine. Since not all engines need to be installed (a HLR server usually has no `paEngine`) each of the structures also contains a string `engines` that will be filled with abbreviations indicating the installed engines (see the description of the CLI tool `glite_dgas_pingClient` in Section 3.1.1 for an example).

If no error occurs, the answer of the HLR server (in XML format) will be contained in `server_answer` and the function's return value will be 0. In case of an error one of the error codes defined in

`interface/glite/dgas/common/hlr/hlr_prot_errcode.h`

will be returned instead.

3.2.2 CONTACTING THE JOBAUTHENGINE

The header file of the client that is used to authorize the accounting transactions for a job in advance is

`interface/glite/dgas/hlr-clients/job_auth/jobAuthClient.h`

It defines a structure that contains the information regarding the job to authorize on the User HLR (see Section 1.3.1), as well as a function that is invoked to contact the HLR server:

```

struct jobAuth_data
{
string dgJobId;
string usrCert;
int submTime;
};

int dgas_jobAuth_client(string &user_acct_bank_id ,
                       jobAuth_data &auth_data,
                       string *server_answer);

```

The `user_acct_bank_id` is the contact string (see Section 3.1.1 for how to form a contact string) of the User HLR.

The unique job ID `dgJobId` must be specified by the caller, while the time of job submission (timestamp) `submTime` can always be retrieved by the `dgas_jobAuth_client` function. `usrCert` is retrieved by the HLR server if GSI communication is enabled at configuration time (that is, if it is known from the security context; this should be the default since an advance authorization is meaningful only if a valid user proxy certificate is required).

If no error occurs, the answer of the HLR server (in XML format) will be contained in `server_answer` and the function's return value will be 0. In case of an error one of the error codes defined in

`interface/glite/dgas/common/hlr/hlr_prot_errcode.h`

will be returned instead.

3.2.3 SENDING USAGE RECORDS TO THE ATMENGINE

Since the transmission of usage records from the Computing Elements to the HLR service are tightly bound to the DGAS software installed on the Computing Elements — that is, the *Gianduia* daemon that collects the usage records and the *CE_Pushd* that sends them to the User HLR — the client APIs²¹ for the ATMEngine are described in [3].

4 INSTALLATION

The following is a brief description of the installation process for both the HLR client programs (usually installed on a User Interface node) and the HLR server that manages the user and resource accounts.

4.1 INSTALLATION OF THE HLR CLIENT PROGRAMS

The HLR clients are usually installed on a User Interface (UI) node. This section treats only the part of the installation process that involves the HLR client components and its dependencies. For more information on the installation of a UI node see the UI installation guides on [10].

The HLR clients, as all gLite components, should be installed on a machine running Scientific Linux CERN (currently version 3, see [7]).

The installation process can be divided into the following steps²²:

²¹Actually there are two APIs: one deprecated, maintained for backward compatibility, and a new one implementing the GGF Usage Record standard for exchanging usage records.

²²Note that the version numbers mentioned in this document may differ from the current version numbers.

1. Make sure that the following external globus packages (external packages for gLite on SLC3 can be found at [8]) are installed:

```

gpt
vdt_globus_essentials
vdt_globus_info_essentials

```

2. Make sure the CA certificates you need are installed (you may want to install all files beginning with “ca_*” from [8]).
3. Install the HLR clients (and the gLite components they depend on) using the rpms from the latest available build (download them from [9]):

```

glite-wms-utils-exception
glite-wms-utils-jobid
glite-wms-utils-tls
glite-dgas-common
glite-dgas-hlr-clients

```

4.2 INSTALLATION OF THE HLR SERVER

The HLR server is usually installed on a dedicated machine although it may be installed together with other gLite services (such as the DGAS PA service). This section treats only the part of the installation process that involves the HLR server components and its dependencies.

The HLR server, as all gLite components, should be installed on a machine running Scientific Linux CERN (currently version 3, see [7]).

The installation process can be divided into the following steps²³:

1. Make sure that the following external globus packages (external packages for gLite on SLC3 can be found at [8]) are installed:

```

gpt
vdt_globus_essentials
vdt_globus_info_essentials

```

2. Make sure the CA certificates you need are installed (you may want to install all files beginning with “ca_*” from [8]).
3. Install and configure a MySQL server and a MySQL client (as all external packages for gLite on SLC3 they can be found at [8]):

```

MySQL-server-4.1.11
MySQL-client-4.1.11
MySQL-shared-compat-4.1.11

```

4. Install the following perl libraries (required for the conversion of DGAS accounting records to APEL LcgRecords with `glite-dgas-convertDGAS2APEL.pl`, see Section 3.1.4):²⁴

²³Note that the version numbers mentioned in this document may differ from the current version numbers.

²⁴This step may be omitted if accounting records will not be forwarded to APEL.

```
perl-DBI
perl-DBD-MySQL
```

5. Install `ntp` for the synchronization of your host to a time server,
6. Install `mkgridmap` (download it from [8]), required to import HLR user accounts from multiple VO LDAP and/or VOMS servers (see Section 3.1.2):

```
glite-security-mkgridmap-2.4.2-1
glite-security-mkgridmap-conf-2.4.2-1
```

7. Install the HLR server (and the gLite components it depend ons) using the rpms from the latest available build (download them from [9]):

```
glite-wms-utils-exception
glite-wms-utils-jobid
glite-wms-utils-tls
glite-dgas-common
glite-dgas-pa-clients
glite-dgas-hlr-service
```

8. Not necessary for an HLR server, but recommended:

```
glite-dgas-hlr-clients
```

5 CONFIGURATION

5.1 PRELIMINARY CONFIGURATION

Running the HLR server correctly requires some specific system configuration as well as a proper configuration of the security environment and the MySQL server.

5.1.1 SECURITY ENVIRONMENT FOR THE HLR SERVER

Make sure that your server has a `grid-mapfile` in `/etc/grid-security`. The same directory has to contain the host key/cert pair. Set the following permissions:

```
chmod 600 /etc/grid-security/hostcert.pem
chmod 400 /etc/grid-security/hostkey.pem
```

If the HLR server is used as both Resource HLR (resource accounts) and User HLR (user accounts): make sure that the subject of the host's own certificate is contained in the `grid-mapfile`, having a line such as

```
"/C=IT/O=INFN/OU=Host/L=Milano/CN=grid003.mi.infn.it" nobody
```

In case of doubt whether the HLR will be used for both, resource and user accounts, or not: add the line!

5.1.2 SYNCHRONIZING THE HLR HOST TO A TIME SERVER

For a correct validation of certificates, required for secure communication and authentication, the system time should be as accurate as possible. It is therefore advisable to synchronize the HLR server machine to a time server using `ntp`. Please see the `ntp` documentation for information on configuration. Make sure the `ntpd` will be started at boot time!

5.2 HLR CONFIGURATION FILE

In the following, the keyword `MANDATORY` means that the parameter needs to be manually inserted in order for the system to work. The keyword `OPTIONAL` means that the system will work even if the parameter is omitted. The keyword `DEFAULT` means that the parameter doesn't need to be modified for the system to work. It can, however, be used to fine tune the system.

The HLR service configuration file is usually named:

```
$GLITE_LOCATION/etc/dgas_hlr.conf
```

and has the following content:

```

<...>
# next parameters set up the conection to the SQL database server

hlr_sql_server = "localhost"
hlr_sql_user = "root"
hlr_sql_password = ""
hlr_sql_dbname = "hlr"
hlr_tmp_sql_dbname = "hlr_tmp"

# gridmap file

gridmapFile = "/etc/grid-security/grid-mapfile"
hostProxyFile = "/tmp/hostProxyFile"

# hlr user: user used to run the daemons

hlr_user = "root"

# default server listening port

hlr_def_port = "56568"

#directory for dgas log and locks

dgas_var_dir = "/opt/glite/var/dgas"

# default log file

hlr_def_log = "/opt/glite/var/dgas/log/hlrd.log"
hlr_qmgr_def_log = "/opt/glite/var/dgas/log/hlr_qmgrd.log"

```

```

#default lock file

hlr_def_lock = "/opt/glite/var/dgas/hlr.lock"
hlr_qmgr_def_lock = "/opt/glite/var/dgas/hlr_qmgrd.lock"
hlr_had_def_lock = "/opt/glite/var/dgas/hlr_had.lock"

#configuration options for the transaction manager daemon.

#expiration period (in second) for a transaction in the queue,
#after this time the priority of the transaction is lowered. DEFAULT 600
hlr_qmgr_expPeriod = "600"

#Number of levels in the queue. transactions enter the queue with
#priority 0 and are increased when the system can't process it.
#priority = hlr_qmgr_qDepth. DEFAULT 10
hlr_qmgr_qDepth = "10"

#Number of transaction processed in every iteration of the process.
#The higher the number of transaction, the higher the resource consumption
#of the process. DEFAULT 10
hlr_qmgr_tPerIter = "20"

#Intervall between two groups of transactions
hlr_qmgr_pollPeriod = "30"

thread_number = "5"

```

The following Sections briefly describe the parameters that can be used to tune the HLR service.

5.2.1 DATABASE SPECIFIC PARAMETERS

hlr_sql_server (DEFAULT, if having MySQL on the same machine, otherwise MANDATORY):

The hostname of the server on which the MySQL HLR database is running, usually it will be the machine of the HLR server, but it can also be a remote MySQL installation accessed via TCP socket.

Default: "localhost"

hlr_sql_user (MANDATORY/DEFAULT):

Specifies the MySQL user with which the HLR server will connect to the underlying database. The user needs full access to the databases specified with `hlr_sql_dbname` and `hlr_tmp_sql_dbname`.

Default: "root"

Note: The system works well using the MySQL root user, but it is advised to create a different user (with full privileges for the necessary databases) for the HLR.²⁵

hlr_sql_password (MANDATORY/DEFAULT):

The password used to authenticate the above mentioned user.

²⁵The creation of a MySQL user with the necessary privileges can be done automatically during the creation of the HLR database with `glite-dgas-hlrd-dbcreate`, see Section 5.3.2.

Default: ""

Note: It is not required to set a password, but it is highly recommended, especially if using the MySQL root user for the HLR.

hlr_sql_dbname (DEFAULT):

The name of the database used to store the HLR tables that contain the permanent mission critical accounting information (accounts and usage records).

Default: "hlr"

hlr_tmp_sql_dbname (DEFAULT):

The name of the database used to store temporary HLR information (that is, the queue of accounting transactions from the User HLR to the Resource HLR that have to be processed asynchronously by the transaction manager daemon). Temporary HLR information is stored in a database structure as well in order to grant persistency to the HLR service.

Default: "hlr_tmp"

5.2.2 SECURITY SPECIFIC PARAMETERS

gridmapFile (DEFAULT):

The name of the grid-mapfile, that is the file used by GSI security to establish a map between X509 certificate *DN* and local unix users.

Default: "/etc/grid-security/grid-mapfile"

hostProxyFile (DEFAULT):

Specifies where to store the host proxy file.

Default: "/tmp/hostProxyFile"

5.2.3 DAEMON SPECIFIC PARAMETERS

hlr_user (DEFAULT):

The user used to run the daemons.

Default: "root"

hlr_def_port (DEFAULT):

The TCP port on which the HLR server should listen. Note that the port is an important part of the HLR's contact string (see Section 3.1.1). Thus changing the port will change the servers contact string.

Default: "56568"

thread_number (DEFAULT):

The maximum number of contemporary threads of the HLR server daemon.

Default: "5"

5.2.4 LOG FILES

dgas_var_dir (DEFAULT):

Parent directory in which the HLR's log and lock files will be written. Note: The log and lock file names still need to be specified as absolute file names (including the full path).

Default: `"/opt/glite/var/dgas"`

hlr_def_log (DEFAULT):

The filename for the main HLR log file.

Default: `"/opt/glite/var/dgas/log/hlrd.log"`

hlr_qmgr_def_log (DEFAULT):

The log file for the daemon responsible of asynchronously managing the incoming job accounting requests.

Default: `"/opt/glite/var/dgas/log/hlr_qmgrd.log"`

5.2.5 LOCK FILES

hlr_def_lock (DEFAULT):

The main HLR lock file.

Default: `"/opt/glite/var/dgas/hlr.lock"`

hlr_qmgr_def_lock (DEFAULT):

The lock file for the daemon responsible of asynchronously managing the incoming job accounting requests.

Default: `"/opt/glite/var/dgas/hlr_qmgrd.lock"`

hlr_had_def_lock (DEFAULT):

The lock file for the HAD daemon described in Section 1.3.3.

Default: `"/opt/glite/var/dgas/hlr_had.lock"`

5.2.6 TRANSACTION QUEUE MANAGER SPECIFIC PARAMETERS

hlr_qmgr_expPeriod (DEFAULT):

The time interval, in seconds, after which a transaction in the job accounting queue (database for temporary storage of information that still has to be processed) is lowered in priority.

Default: `"600"`

hlr_qmgr_qDepth (DEFAULT):

The number of priority levels in the job accounting queue. When an accounting transaction (that starts with priority 0) has been retried without success until its priority exceeds this number, it will no longer be automatically processed and can be considered in "permanent error condition" (accounting information is not lost, but remains in the transaction queue and does not appear on user/resource accounts). The priority of a transaction that could not be processed is incremented after the time interval specified with `hlr_qmgr_expPeriod`.

Default: `"10"`

hlr_qmgr_tPerIter (DEFAULT):

The maximum number of accounting transactions processed at each iteration of the process. The higher the number of transactions, the higher the resource consumption of the process.

Default: “20”

hlr_qmgr_pollPeriod (DEFAULT):

The time interval, in seconds, between two iterations of transaction processing. The lower the interval, the higher the resource consumption of the process.

Default : “30”

5.3 CONFIGURATION OF THE HLR DATABASE

The following describes how to configure the MySQL server and create the HLR database.

5.3.1 CONFIGURATION OF THE MYSQL SERVER

Configure MySQL not to listen for remote connections, if installing a dedicated DGAS HLR for which only local access to the database is required: Make sure that the line “skip-networking” is specified under the configuration group “[mysqld]” in the configuration file /etc/my.cnf. This is especially important if no password is used for MySQL (see parameters `hlr_sql_user` and `hlr_sql_password` of the HLR configuration file, see Section 5.2.1).

Important notes:

- Make sure the `mysqld` will be started at boot time!
- If no MySQL root password was set so far (default after installation of the MySQL server), set a new root password, especially if the server is configured to listen for remote connections.

5.3.2 CREATION OF THE HLR DATABASE

This step will create the HLR databases and the MySQL user (if it does not yet exist) with the necessary privileges. The MySQL user and databases are those specified in the HLR configuration file (see Section 5.2.1).

A single MySQL user may be used for both HLR and DGAS Price Authority (PA) if they are installed on the same host.

Launch the script that creates the databases and the MySQL for the HLR:

```
$GLITE_LOCATION/sbin/glite-dgas-hlr-dbcreate <mysqlRootPassword>
```

If the MySQL root password is not specified on the command line, the script will assume that no password is set. Running this script creates the databases and the MySQL user (if it does not yet exist) specified in the HLR configuration file (see above). In case the MySQL user does already exist the script will check if the password specified in the configuration file is correct.

5.4 CONFIGURATION OF OPTIONAL ADMINISTRATION TOOLS

Two of the optional administration tools need to be configured before they can be used:

5.4.1 CONFIGURATION OF AUTOMATIC USER AND RESOURCE ACCOUNT IMPORT

`glite-dgas-hlr_userAccountImport.pl`, the script used to import user accounts from multiple VOMS and/or LDAP servers, is based on the external tool *glite-mkgridmap* and requires a configuration file similar to the one used by that tool. A detailed description, including an example for the configuration file, can be found in the description of `glite-dgas-hlr_userAccountImport.pl` in Section 3.1.2.

The default configuration file used by `glite-dgas-hlr_userAccountImport.pl` is:

```
$GLITE_LOCATION/etc/glite-dgas-hlr_mkgridmap_userAccountImport.conf
```

The other scripts for account import, `glite-dgas-hlr_voImport.pl` (for user account import from a single LDAP server; see Section 3.1.2) and `glite-dgas-hlr_bdiiImport.pl` (for resource account import from a single BDII server; see Section 3.1.2) do not require any configuration, but need to be run with appropriate command line arguments.

We recommend to run the scripts periodically (e.g. as cron jobs; at least once a day) in order to import new user accounts when they are added to the queried VOMS or LDAP servers. The created log files might also require a backup (e.g. with `logrotate`). A balance between a frequent creation of user accounts and a small overhead for the HLR host should be found. We suggest to run the import tool once a day. Alternatively, the HLR administrator may also add user accounts manually whenever they are required.

5.4.2 CONFIGURATION OF CONVERSION OF ACCOUNTING RECORDS FROM DGAS TO APEL

`glite-dgas-convertDGAS2APEL.pl`, the script used to convert DGAS accounting information to the `LcgRecords` entries used by APEL [11], requires access to different databases. A detailed description of the script and its database access requirements, including an example for the configuration file, can be found in the description of `glite-dgas-convertDGAS2APEL.pl` in Section 3.1.4.

The default configuration file used by `glite-dgas-convertDGAS2APEL.pl` is:

```
$GLITE_LOCATION/etc/glite-dgas-convertDGAS2APEL.conf
```

In case a conversion to APEL's `LcgRecords` should be required, we recommend to run the script periodically (e.g. as a cron job; once a day). The created log file might also require a backup (e.g. with `logrotate`).

6 RUNNING THE HLR SERVER

The HLR service consists of three daemons:

- *hlr_serverd*: The service that is listening for incoming connections and loads the engines that process them.
- *trs_mgrd*: The transaction manager that periodically processes the usage records/transactions in the queue (see the description of the HLR configuration file in Section 5.2.6)
- *HAD*: The High Availability Daemon that is described in Section 1.3.3.

The three daemons required for the HLR server can be controlled with a single start-up script:

```
$GLITE_LOCATION/sbin/glite_dgas_hlrd
```

Available options are:

```

> $GLITE_LOCATION/sbin/glite_dgas_hlrd -h
Usage: glite_dgas_hlrd {start|stop|restart|status}

> $GLITE_LOCATION/sbin/glite_dgas_hlrd start
Starting hlr_serverd:           [ OK ]
Starting trs_mgrd:             [ OK ]
Starting DGAS HAD...          [ OK ]
  
```

This command does control all daemons necessary for the correct functioning of the HLR server, that is, not only the HLR server daemon itself, but also the transaction manager daemon, that asynchronously handles transactions between HLR servers, as well as the High Availability Daemon, that monitors the other two daemon's status (see Section 1.3.3).

To verify if the HLR server daemon is running and listening on the port specified in the configuration file (default port is 56568 as used in the example below) simply type:

```

> netstat -tln | grep 56568
tcp        0      0  *:56568                *: *                    LISTEN
  
```

7 KNOWN PROBLEMS AND CAVEATS

Most problems arise from bad configuration of the HLR server or from missing dependencies. Make sure that you installed all the packages mentioned in Section 4 and followed all the necessary configuration steps described in Section 5. Missing certificates (host certificates, as well as CA certificates) for example will allow the server to run, but the missing security context will not allow it to be contacted from remote hosts.

For precaution, a secondary daemon, called *HAD*, has been added to periodically verify the HLR server status and eventually restart it if necessary. Moreover, a server that is temporarily unavailable does not necessarily imply a loss of usage records or accounting transactions, since both the CE Pushd daemon [3] — that sends the usage records from the Computing Element to the User HLR — and the User HLR's transaction manager — that sends the accounting information (and eventual payment) to the Resource HLR —, retry several times²⁶ in case of connection failures.

REFERENCES

- [1] I. Foster et al. "A Security Architecture for Computational Grids". In *Proc. of the 5th ACM Conference on Computers and Security*, pp. 83-91, ACM Press, 1998.
- [2] A. Guarise, G. Patania and R.M. Piro. "DGAS - Price Authority. User's Guide". Technical Report EGEE-JRA1-TEC-571271-PA. August 5, 2005.
- [3] A. Guarise, G. Patania and R.M. Piro. "DGAS - GIANDUIA. User's Guide". Technical Report EGEE-JRA1-TEC-571271-GIANDUIA. August 5, 2005.

²⁶The number of tentatives and the time between tentatives can be set in the configuration files. For the transaction manager see Section 5.2.6.

- [4] R.M. Piro, A. Guarise, and A. Werbrouck. "An Economy-based Accounting Infrastructure for the DataGrid". *Proceedings of the 4th International Workshop on Grid Computing (GRID2003)*, held in conjunction with the SC2003 Conference, Phoenix, Arizona, November 17, 2003.
- [5] R.M. Piro, A. Guarise, and A. Werbrouck. "Simulation of Price-sensitive Resource Brokering and the Hybrid Pricing Model with DGAS-Sim". *Proceedings of the 13th International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2004)*, track on Emerging Technologies for Next Generation Grid (ETNGRID 2004), Modena, Italy, June 14-16, 2004.
- [6] R.M. Piro, A. Guarise, and A. Werbrouck. "Price-sensitive Resource Brokering with the Hybrid Pricing Model and Widely Overlapping Price Domains". Accepted for publication in a special issue of *Concurrency and Computation: Practice and Experience* (Wiley Publishers).
- [7] Scientific Linux CERN 3 (SLC3). <http://linux.web.cern.ch/linux/scientific3/>
- [8] External packages for gLite on SLC3. <http://glite.web.cern.ch/glite/packages/externals/bin/rhel30/RPMS/>
- [9] Latest build of gLite packages. <http://glite.web.cern.ch/glite/packages/#latest.builds>
- [10] JRA1 Workload Management website. <http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/>
- [11] R. Byrom et al. LCG Accounting Schema. October 2004. <http://goc.grid-support.ac.uk/gridsite/accounting/apel-schema.pdf>