



DataGrid

DATA GRID ACCOUNTING SYSTEM ARCHITECTURE - V 1.0



Document identifier: **DataGrid-01-TED-0126-1_0**

Date: **14/02/2003**

Work package: **WP1**

Partner: **INFN**

Document status **DRAFT**

Deliverable identifier:

Abstract: A framework for accounting in the DataGrid Project is presented as a series of services each with well specified functionality and sufficient flexibility to operate with commonly contemplated economic models. This revision of the document is the first concerning EDG 2.0



Delivery Slip

	Name	Partner	Date	Signature
From	A.Guarise	INFN		
Verified by				
Approved by				

Document Log

Issue	Date	Comment	Author
0.8	01/07/2002	First public version	A.Guarise, A.Werbrouck
1.0	14/02/2003	Revised to comply with EDG 2.0	A.Guarise, R.Piro

Document Change Record

Issue	Item	Reason for Change

Files

Software Products	User files
Word	DataGrid-01-TED-0126-0_0-acct_arch.doc
Acrobat	DataGrid-01-TED-0126-0_0-acct_arch.docpdf



CONTENT

1. INTRODUCTION	4
1.1. OBJECTIVE OF THIS DOCUMENT	4
1.2. APPLICATION AREA.....	4
1.3. TERMINOLOGY.....	4
2. ACCOUNTING PURPOSE AND FEATURES	6
2.1. PURPOSE	6
3. WORKING SCHEME	8
3.1. NOTATION.....	8
3.2. JOB SUBMISSION.....	8
4. PREMISES.....	10
5. FLOW-CHARTS.....	11
5.1. INTERFACES FOR THE ACCOUNTING SYSTEM.....	11
5.2. STRUCTURE OF THE SERVER DAEMON.....	13
5.3. BANK SERVICE	13
5.4. UI – JOBAUTH SERVICE.....	18
5.4.1. <i>JobAuth service flowcharts</i>	18
5.4.2. <i>UI - JobAuth service interaction diagrams</i>	20
5.4.3. <i>UI-JobAuth service deployment diagram</i>	21
5.5. RB – HLR USERAUTH SERVICE	22
5.5.1. <i>HLR userAuth service flowcharts</i>	23
5.5.2. <i>RB - HLR service interaction diagrams</i>	25
5.5.3. <i>RB-HLR service deployment diagram</i>	26
5.6. PRICE AUTHORITY SERVICE.....	27
5.6.1. <i>RB - PA service interaction diagrams</i>	31
5.6.2. <i>RB-PA service deployment diagram</i>	31
5.7. USER-HLR UI SERVICE.....	33
5.8. SENSORS – ATM SERVICE	36
5.8.1. <i>ATM Service flowcharts</i>	38
5.8.2. <i>Sensor –ATM_service Interaction diagram</i>	42
5.8.3. <i>Sensor –ATM_service deployment diagram</i>	43
5.9. GENERAL SCHEME	44
5.9.1. <i>Job debiting & crediting calls scheme</i>	44
5.9.2. <i>Job debiting & crediting main flow</i>	45
6. DATABASES DESCRIPTION	49
6.1. BANK SERVICE DATABASE	49
6.2. PRICING AUTHORITY DATABASE	51
7. CONCLUSIONS.....	52
8. ANNEXES.....	53
8.1. SOFTWARE DESCRIPTION	53

1. INTRODUCTION

1.1. OBJECTIVE OF THIS DOCUMENT

The objective is to propose an accounting component which is efficient, flexible and compatible with the other already functioning components of the DataGrid WP1 middleware. Here we illustrate the schematics of the accounting software and its main functionalities.

For further information see [R1]

1.2. APPLICATION AREA

Reference documents

[R1] C.Anglano et al. – *An accounting system for the DataGrid Project v3.0* - DataGrid-01-TED-0115-3_0 – http://www.to.infn.it/grid/accounting/Current_prop.pdf

1.3. TERMINOLOGY

Definitions

Computational Energy	A quantity expressing the total amount of computational effort expended by a job's execution.
Resources	Elements which are needed for a job's execution.
GridCredits	Currency used in the economic transactions between producers (the computing resources) and consumers (the Grid users) on the Grid.
HLR	DataBase that maintains the fund status of users and resources; it is also responsible for managing the economic transactions between producers and consumers. There will be many HLRs spread over the Grid, each HLR will manage a subset of the Grid users and/or resources.
Job cost	The cost of a job's execution in GridCredit units.
PA: Price Authority	An authority that can set the usage prices of the resources.
Resource value	An estimate of the real contribution of this resource to job executions.
Resource price	The price assigned to a resource element starting from its value.
VO: Virtual Organization	An organization that administratively groups a set of user and/or resources.
ATM	Automatic Transaction Manager is the element in this accounting infrastructure that manages the crediting and debiting process.

Glossary

Economic model	A model that regulates the virtual economic transactions between all the entities involved in the Grid.
Authentication	The phase in which a user and a resource mutually check each other's identities.
Authorization	The phase in which the system grants to the user access to the system
Accounting	The phase in which the system audits the usage of system resources.



QTT

Queue Traversal Time

GMT

Greenwich Mean Time



2. ACCOUNTING PURPOSE AND FEATURES

2.1. PURPOSE

The DataGrid Accounting System (DGAS) is meant to furnish a distributed accounting infrastructure to the DataGrid Project.

The DGAS software aims to furnish to the EDG users an economic-based accounting environment. This means that from the DGAS perspective both user and resources are seen as entities capable of exchanging "virtual credits" that we usually call GridCredits. The most usual case representative for this exchange is the that of a user submitting a job to a grid-resource: the user will pay to the resource a well defined amount of GridCredits in order to get his job executed. Generally a user shall receive the amount of GridCredits needed to perform his computations by the management of the research-group he belongs to.

It should be straightforward to see that the research groups will have their own grid-resources and that these resources will earn credits by executing the user jobs. These credits can then be redistributed among the users belonging to that group.

It's believed that such approach has the following advantages over a standard accounting system (in which the system keeps a flat record of the resource consumption by each user):

1) It is user friendly, since everybody is familiar with basic economic concepts.

In fact while it can be difficult to understand at a glance the usage consumption of a job analysing info such as cpu time, resident memory used, disk storage and similar technical details, it's easy to understand if the effort required by a job was small or big by means of the amount of "money" required to get that job executed. This approach however still permits the user or the system administrator to know the details of the computations.

2) It's easy to avoid indiscriminate usage of the grid by single users.

Since every user has a well defined amount of credits available, he won't submit more jobs than what he can afford thus minimizing the risk of users saturating the grid with job-trials (or worse with deliberate abuse of grid-resources).

3) It can be used to help the Workload Management process by means of natural economic feedbacks.

This aspect is also known as Economic Brokering. The idea is that an economic accounting environment naturally creates an "exchange market" where the users wants to maximize their computing capabilities while minimizing their expenses while their counterparts, the resource owners, want to maximize their earns while minimizing their expenses. So the resource owners (or a suitable automatic system) will manage the resource prices in order to minimize the idle time of the CPUs (usually lowering the prices for idle resources), while the user will usually seek for the cheapest resources. This is expected to generate a feedback that, if well tuned, should result in an equilibrium state where there aren't to many idle resources, so maximizing the grid throughput.

4) Motivates the organization involved in the Grid to share their computing resources.



This because the more resources you have, the more credits you earn.

5) Makes it feasible (where desired) to involve third-parties such as big computing centers or industrial partners, since it is easy to exchange computing power by means of GridCredits.

There may be entities not interested in a pure exchange of computing power, an example may be the big computing center that usually has huge amount of computing power but not enough user to keep its CPUs busy.

The DGAS software implements both an Economic Accounting system and a way to implement an Economic Brokering experiment.

The Economic Accounting system is composed by a set of geographically distributed servers, named HLR (Home Location Register) that can be seen as sort of bank-branches where the accounts for both user and resources resides. There be many way to distribute the HLR over the Grid, thus partitioning the users and the resources. The one that we propose is to have one HLR per Virtual Organization. It should be clear that in principle many other topologies are feasible.

The working scheme is simple: Once a user submits a job, the WMS checks on the user HLR if he as a valid account, if so it proceeds with the submission. Once a job has concluded, a process running on the GridResource collects some information about the job such as the CPU_TIME consumed. These info, commonly called Usage Records are the sent to the User HLR, where are used to compute the job cost. At this point the User HLR sends a credit-ticket to the resource HLR, where the resource account is credited for that amount of credits, and then on the user HLR the user account is debited of the same amount.

In order to work, this system needs that the user specifies the address of his HLR in the JDL of the Job and that in the GRIS of the resource there must also be a field specifying the address of the resource HLR.

The Economic Brokering infrastructure is implemented as follow: every resource can be "linked" to a service called Price Authority (PA). This service is responsible of assigning a price for the resource usage. The PA assigns a price to a resource using the information about the status of its queues. The idea is to assign the prices in order to incentive the user (or the broker on their behalf) to submit jobs on free resources. Since the price-algorithm is implemented as a DLL it's easy to modify the behaviour of the system by changing the algorithm itself. In order for this system to work every resource should publish in its GRIS a field containing the address of its PA.

It is clear that these services need a tight security in order to be reliable, clearly the most important aspect is that every communication to and from the DGAS services must be mutual authenticated and encrypted, for this reason the communication layer is built on top of the Globus GSI libraries.

3. WORKING SCHEME

3.1. NOTATION

In order to correctly describe our accounting system we will use the following terms:

- **K(i,j)**: j-th entity associated with the organization i, K stands for one of the following types of entity.
 - **U**: User.
 - **R**: Computing Resource.
 - **J**: Job.
- **HLR**: Home Location Register.
- **PA**: Price Authority.
- **UI**: User Interface.
- **RB**: Resource Broker.
- **IS**: Grid Information Service.
- **JC**: Job Controller.
- **WMS**: Workload Management System.

Example: U(5,3) stands for the user number 3 of the organization identified by the id 5.

Now we can examine the working scheme of our system when a user submits a job to the grid.

3.2. JOB SUBMISSION

Here we present how the DGAS software intervenes in the job submission procedure. Some of the detail presented can be slightly different from the behaviour of the components foreseen for the first integration of the software. These differences will be highlighted where necessary.

- **A**: The user U(i,j) submits a job to the WMS via the UserInterface using the JDL specification language. The request should contain those parameters needed by the RB in order to optimally choose the set of resources, and from those resources to compute the job cost. The User Interface then, on behalf of the User must send to his HLR a ticket to authorize the payment of the job.
- **B**: The Information Service provides to the RB a complete list of resources that satisfy the job requirements. Then the RB obtains the price list for these resources from the Price Authorities and uses a Cost Estimation Algorithm to optimise the job cost¹. The estimation will be done according to the information given by the user in the JDL.
- **C**: The RB asks the HLR of the user whether the latter's funds are adequate.
- **D**: The HLR responds to the previous enquiry. If the answer is affirmative, that amount of fund is reserved, if not the Broker must intervene.
- **E**: The job can then be submitted to the Computing Element.

¹ Up to now the cost estimation algorithm is unavailable. We defined only the interface of the API, but it is not currently feasible to obtain the needed information from the user (such as CPU time consumed by the job, storage needed and so on).

- **F:** The job runs on the Computing Element and should be continuously monitored by the Monitor Service.
- **G:** The usage records and job info gathered by the Monitor Service should be periodically sent to the User HLR. On the User HLR the Cost Algorithm uses this information to compute the partial cost for that job. If the cost exceeds the amount of funds assigned to that job, the HLR can suspend the execution of the job (however initially the Accounting Infrastructure won't suspend any job, regardless of the cost overrun²).
- **H:** Once the job finishes (satisfactory or not) the user HLR computes its total cost. Finally the user HLR sends to the Computing Element HLR the credits spent by the job and unlocks any residual amount of the funds.

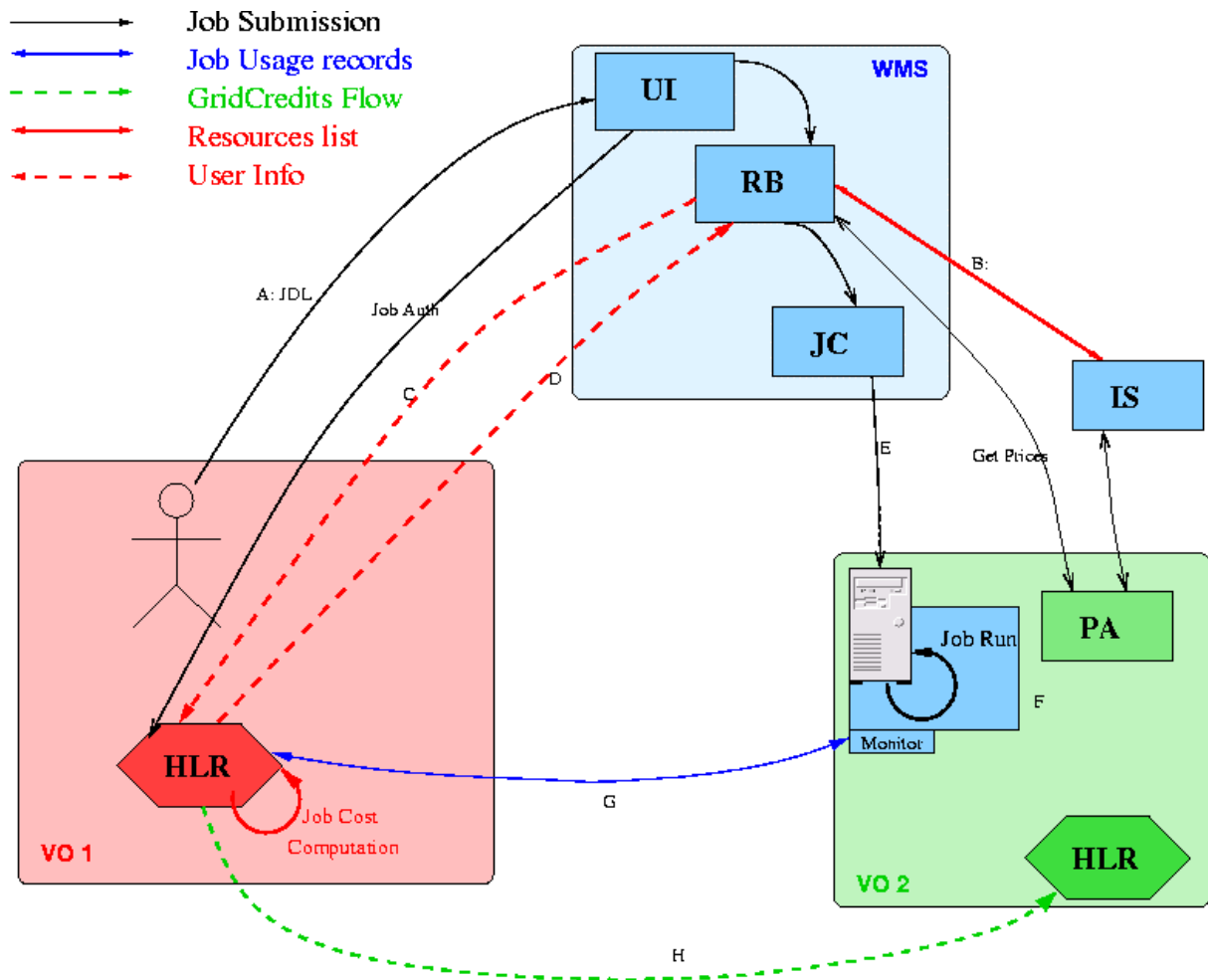


Figure 1: Job submission scheme

² Up to now the sensor system furnishes the usage records only when the job is effectively concluded.



4. PREMISES

In order to correctly identify the DGAS services we define the following attributes:

```
_ACCT_PA_ID = host:port:x509_subject  
_ACCT_BANK_ID = host:port:x509_subject
```

The first is used by the clients to contact the Price Authority-service, the second a general bank service. They both contain the hostname and the port number of the server listening for incoming connections. The last information is the X509 certificate subject of the respective host and is used to enable mutual authentication between the client and the server.

It is important to have mutual authentication to avoid someone “spoofing” a trusted service with a malicious one.

In our context, there can be both user and resource bank services which will be identified by the use of the USER or RES prefixes.



5. FLOW-CHARTS

5.1. INTERFACES FOR THE ACCOUNTING SYSTEM

Several interfaces and associated protocols need to be defined within the accounting system as indicated in Figure 2. These cover communication between users and their HLR, between the Pricing Authority and its associated HLRs, between different HLRs, and among other grid components (Sensors, RB, SE, etc...) and the HLRs that are owners of resources that furnish computational energy.

One expects that several, if not all, HLRs are associated with a single logical Pricing Authority³. The principal part of these protocols are requests for information or actions by a client and a corresponding response by a server.

To permit flexibility and ease of use in these communications it is necessary to choose a powerful and extensible, but also user friendly, language in which to express requests and responses. After an examination of such vehicles, the accounting group chose XML which is already well known on the WEB as a meta language in which information is expressed in such a way that it can easily be visualized using browsers in widely different terminals, both fixed and mobile. The wide diffusion of XML has resulted in free access to excellent parsers, a necessary part of each input port.

The accounting system is composed of a number of services: Bank service (HLR), Price Authority Service, ATM Service and User Interface Service.

The modularity of the architecture should easily permit to add new services to implement new functionalities.

³ There may be physical instances of the same logical PA which however function identically and share a common dataset.

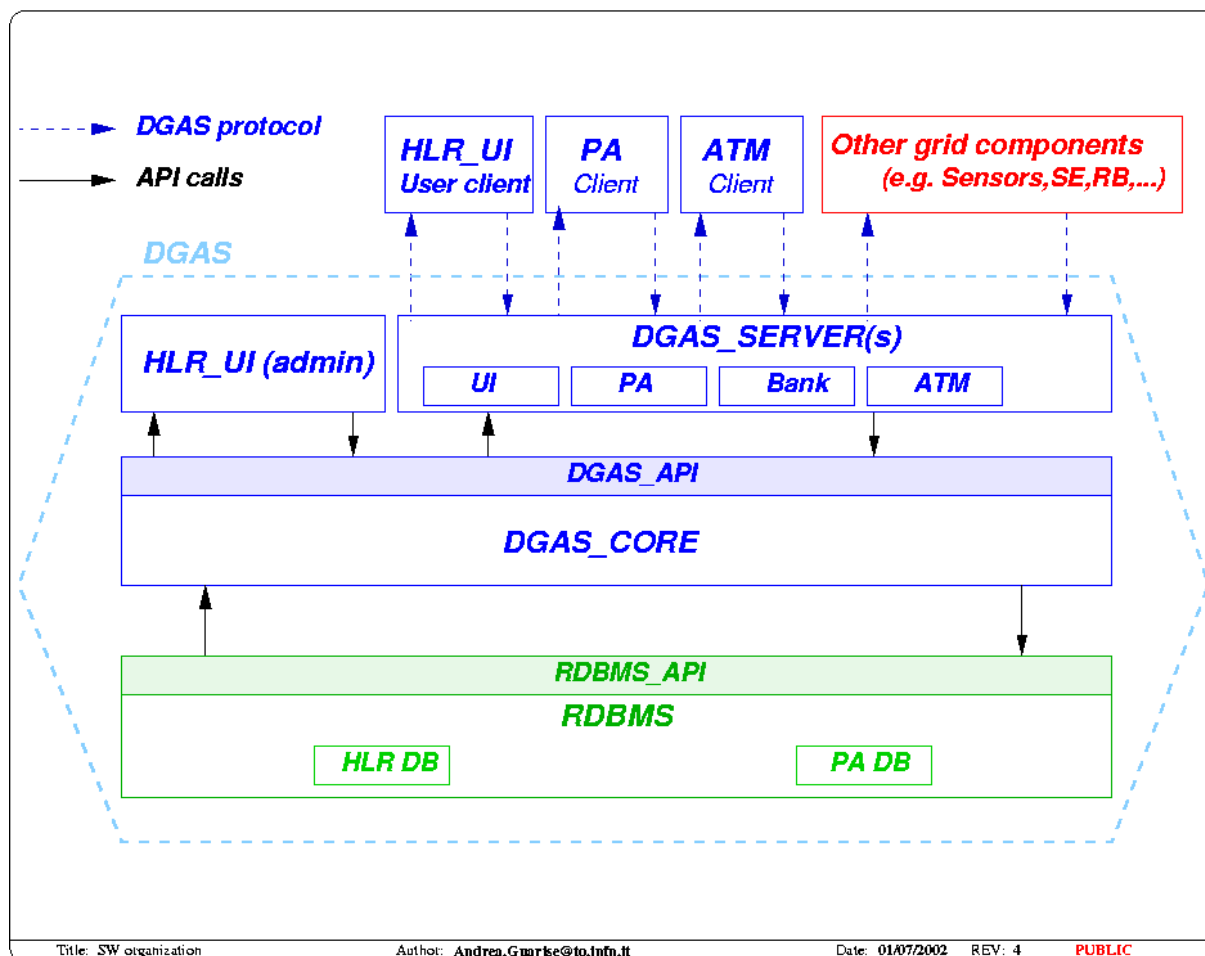


Figure 2 : Interfaces scheme. Dashed lines represent DGAS protocols in XML and solid lines represent calls to Application Programs which must satisfy the AP interfaces. The DGAS_CORE contains several functional modules which interface to the SQL DBMS. One logical DGAS server (light blue dashed line) may contain more than one physical server listening on different TCP ports, e.g. one PA dedicated server and one or more HLR/banking servers, may be sharing the same RDBMS.

5.2. STRUCTURE OF A DGAS SERVER DAEMON

Each service is organized as a client and a server. All calls for a service interface with the client using XML messages. Incoming requests are analysed, combined with other information, and a reply message is prepared by the service. When a failure is signalled by the server the client is responsible for determining whether one or more retries are justifiable before signalling failure to the caller.

Each server has a daemon waiting for an input message and containing an engine specific to the function of the server. Figure 3 illustrates the schematics daemon structure.

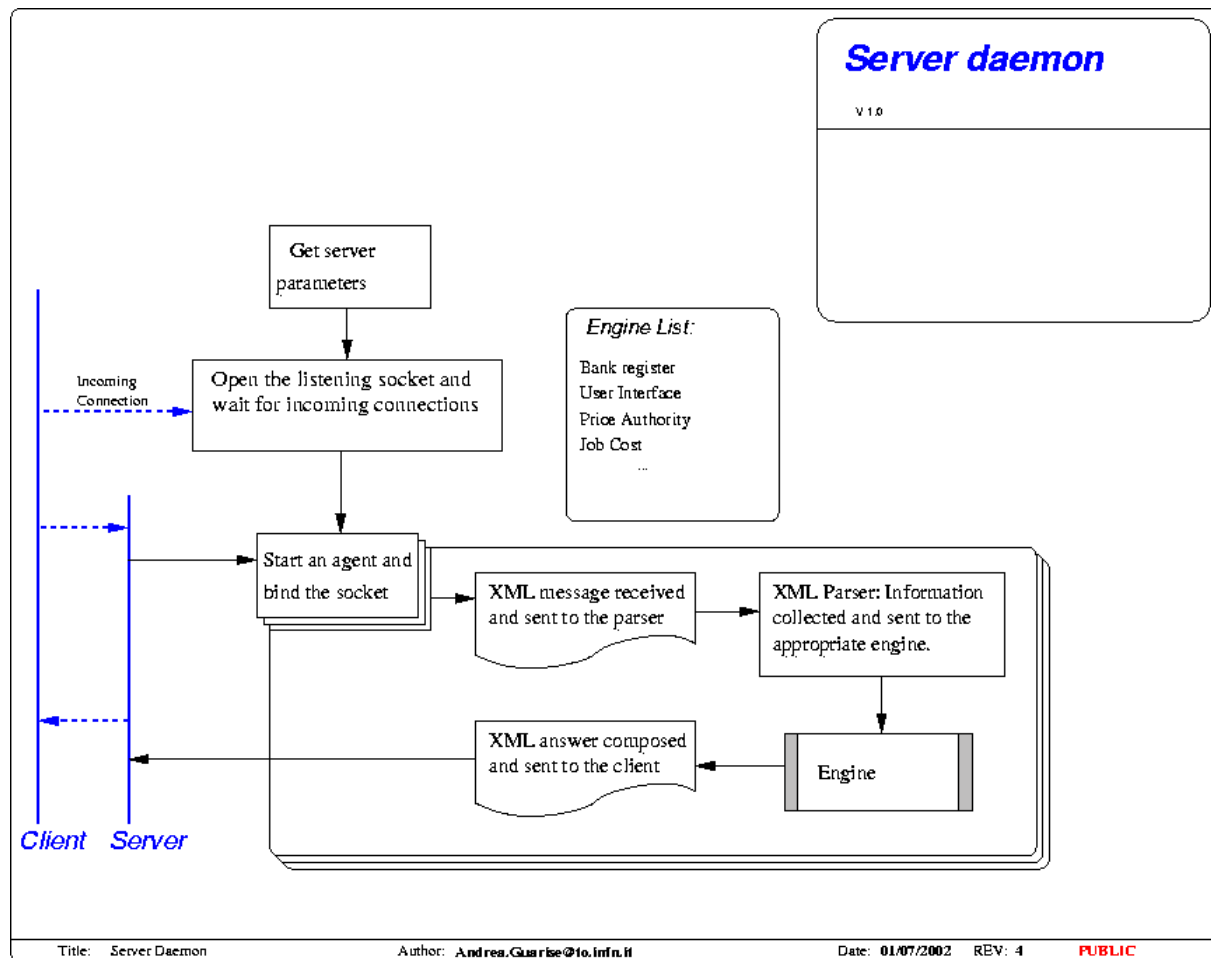


Figure 3: Schematic daemon structure

5.3. BANK SERVICE

Figure 4 and Figure 5 represent the Bank Service client and server respectively. Each distinguishes between the two processes: reserving funds before execution, crediting the resource owner account and debiting the user account after execution.

Since this service operates within a message switching environment, checks are included to avoid to process a single job transaction more than once.

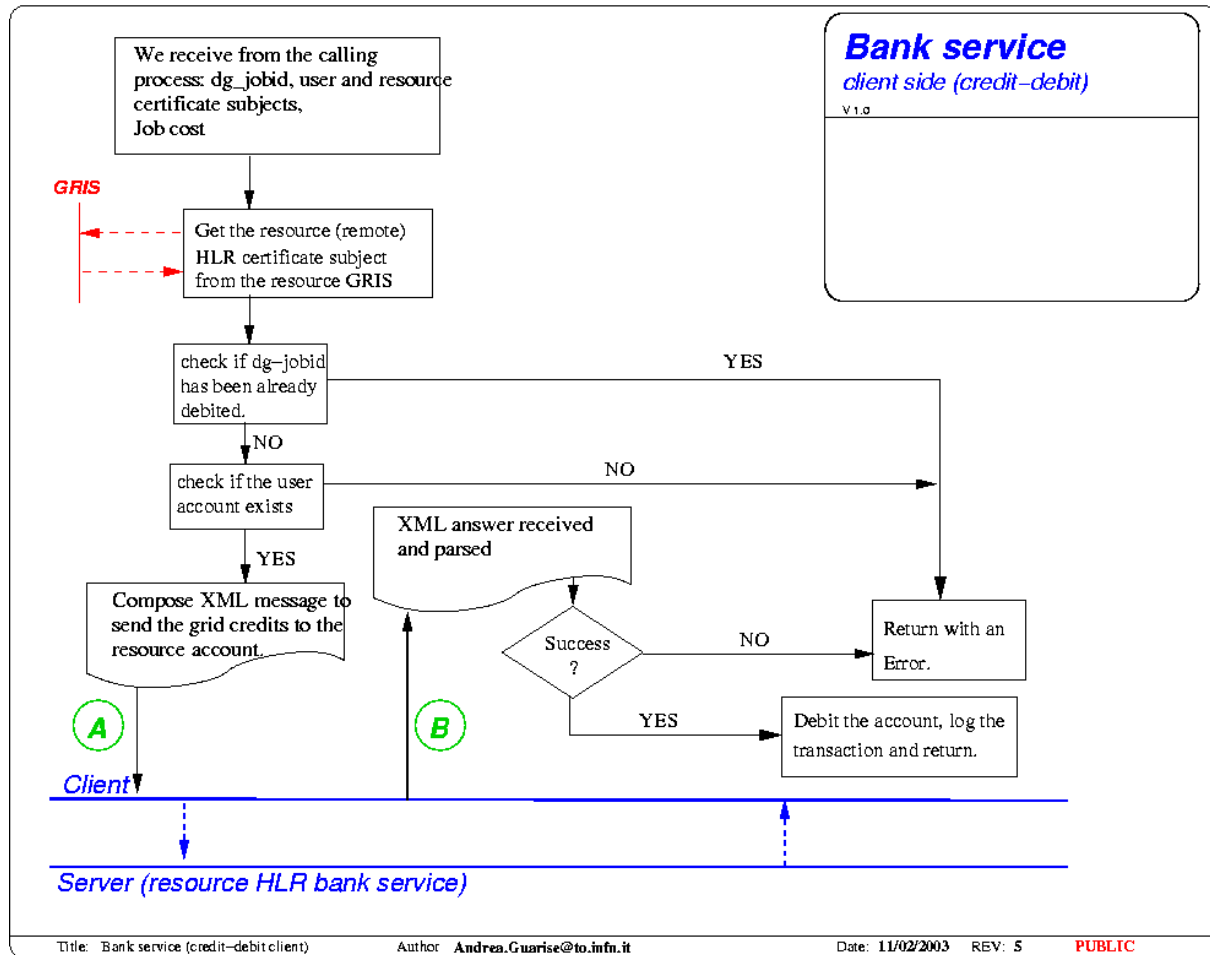


Figure 4: Bank service, client side.

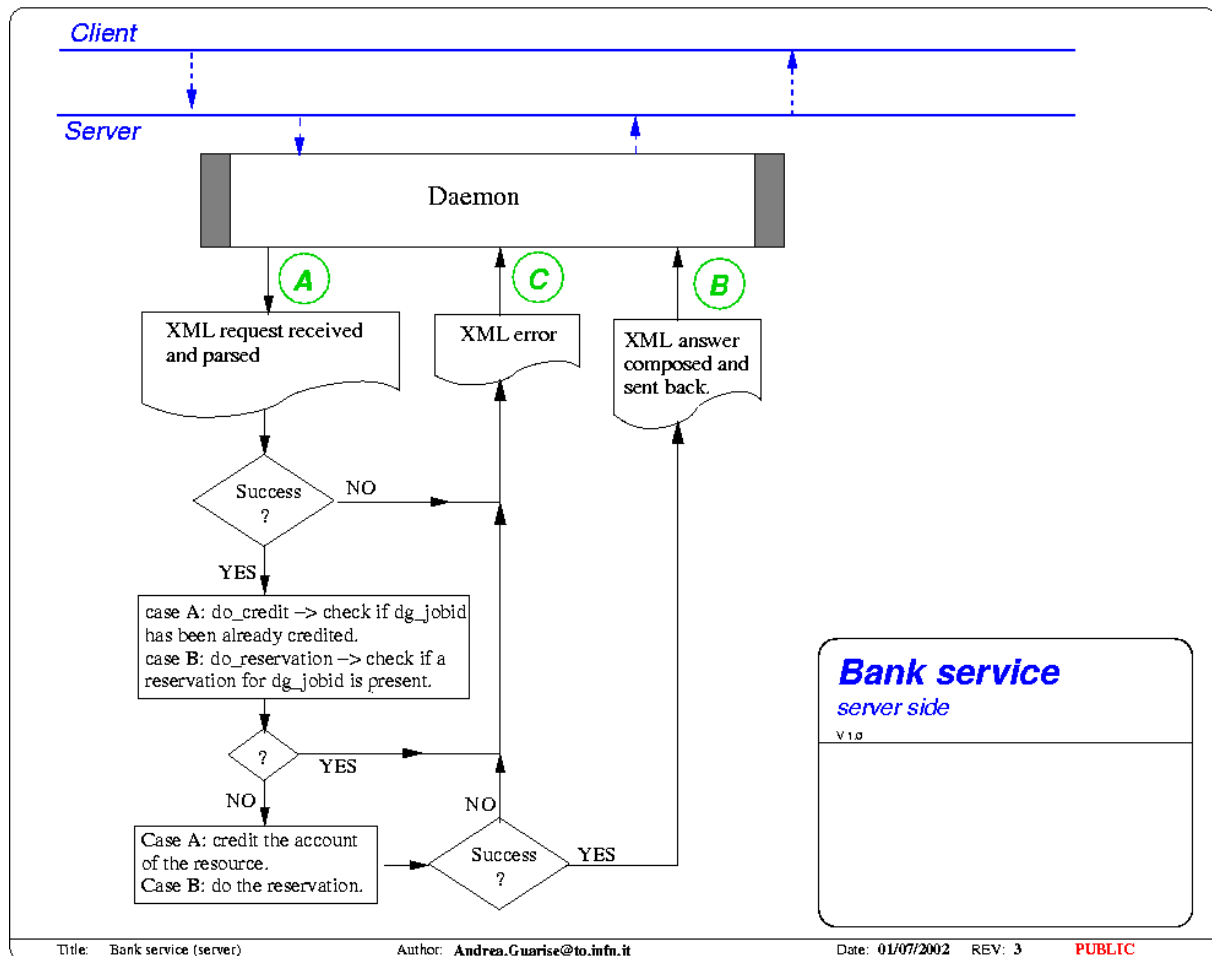


Figure 5: Bank Service, server side.

XML A, Query from the client.

Description: The client (in this case the User HLR) tells the remote HLR (the resource's one) to credit The resource account with the amount of credits consumed by the job.

Message:

```
<HLR type="bank_do_credit">
  <HEAD>
    <VER>
      1.0
    </VER>
  </HEAD>
  <BODY>
    <TR_INFO>
      <TO_GRID_ID>
        Grid id (e.g. the CE_ID) of the account credited
      </TO_GRID_ID>
      <FROM_GRID_ID>
        Grid id (e.g. the user X509 subject) of the remote debited account.
      </FROM_GRID_ID>
    </TR_INFO>
  </BODY>
</HLR>
```



```
</FROM_GRID_ID>
<FROM_HLR_URL>
  URL (hostname:port:contactString) for the remote HLR
</FROM_HLR_URL>
<ID>
  ID for the transaction (e.g. the dgJobID)
</ID>
<AMOUNT>
  Amount to be credited
</AMOUNT>
<LOGDATA>
  Data to be logged with the transaction
</LOGDATA>
</TR_INFO>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: The server part processed the transaction successfully and sends a receipt to the remote HLR.

Message:

```
<HLR type="bank_receipt">
  <HEAD>
    <VER>
      1.0
    </VER>
  </HEAD>
  <BODY>
    <TR_INFO>
      <TO_GRID_ID>
        Grid id (e.g. the CE_ID) of the account credited
      </TO_GRID_ID>
      <FROM_GRID_ID>
        Grid id (e.g. the user X509 subject) of the remote debited account.
      </FROM_GRID_ID>
      <ID>
        ID for the transaction (e.g. the dgJobID)
      </ID>
      <AMOUNT>
        Amount to be credited
      </AMOUNT>
      <STATUS>
        Exit code, 0 success >0 failure.
      </STATUS>
    </TR_INFO>
```



</BODY>
</HLR>

5.4. UI – JOBAUTH SERVICE

The purpose of this service is to inform the accounting system of the existence of a job. When a job is being submitted by the User Interface to the Workload Management System the UI also sends a message to the user HLR server. This message contains both the dg_JobId of the job and the subject of the User X509 certificate. Since the UI commands run with the user privileges, the server can authenticate the user assuring that the job will be debited to its real owner.

The information gathered by this service will then be used by the ATM service, together with the data furnished by the sensor service, in order to process the transaction.

Concurrently with this process, the UI must also check that the USER_BANK_ID given by the user is registered in the list of the DataGrid trusted HLRs. This check is needed because otherwise a malicious user could set up his own fake HLR with falsified accounts thus resulting in forging his own GridCredits.

5.4.1. JobAuth service flowcharts

In Figure 6 and Figure 7 we present the flowcharts of this service.

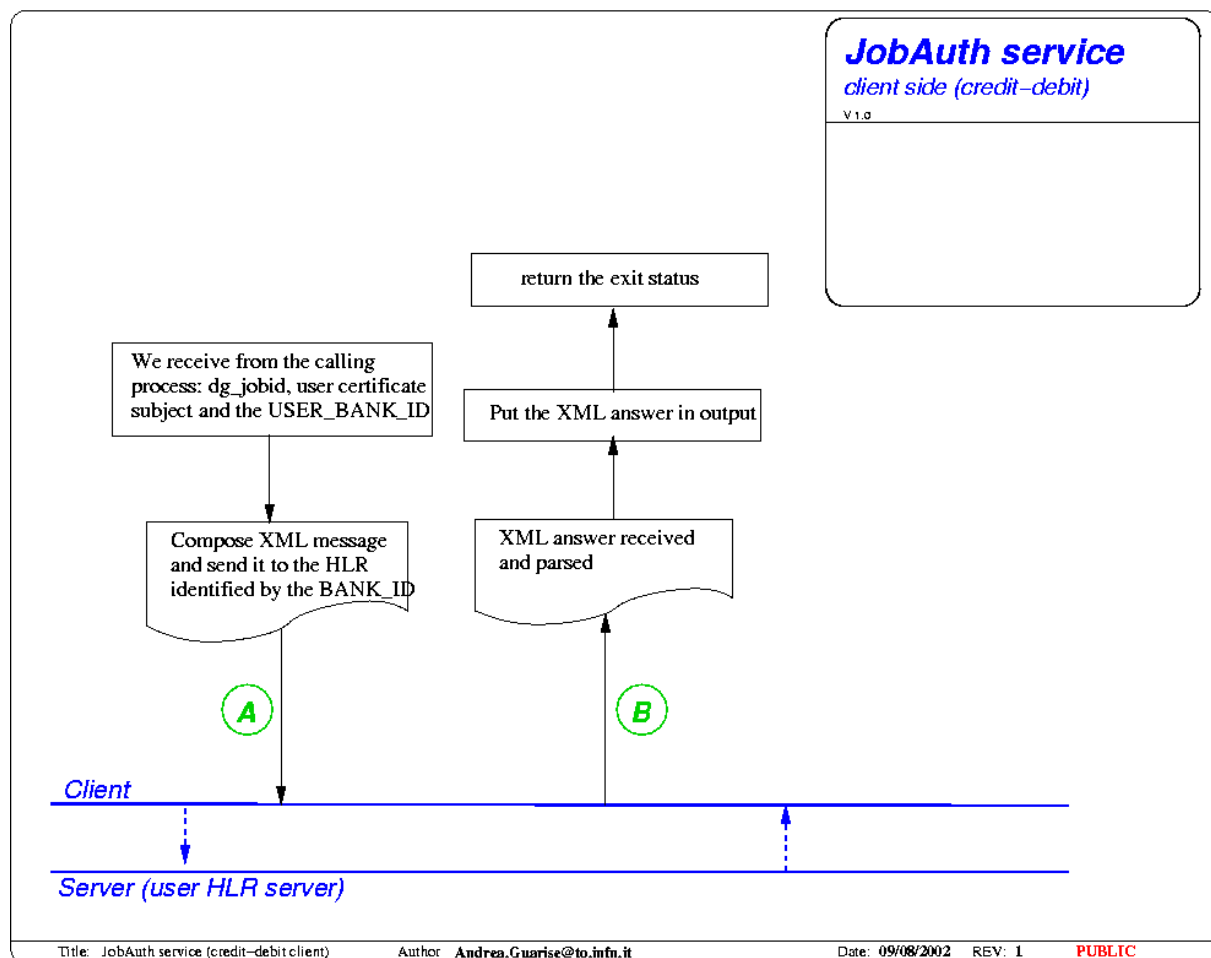


Figure 6: JobAuth client flowchart

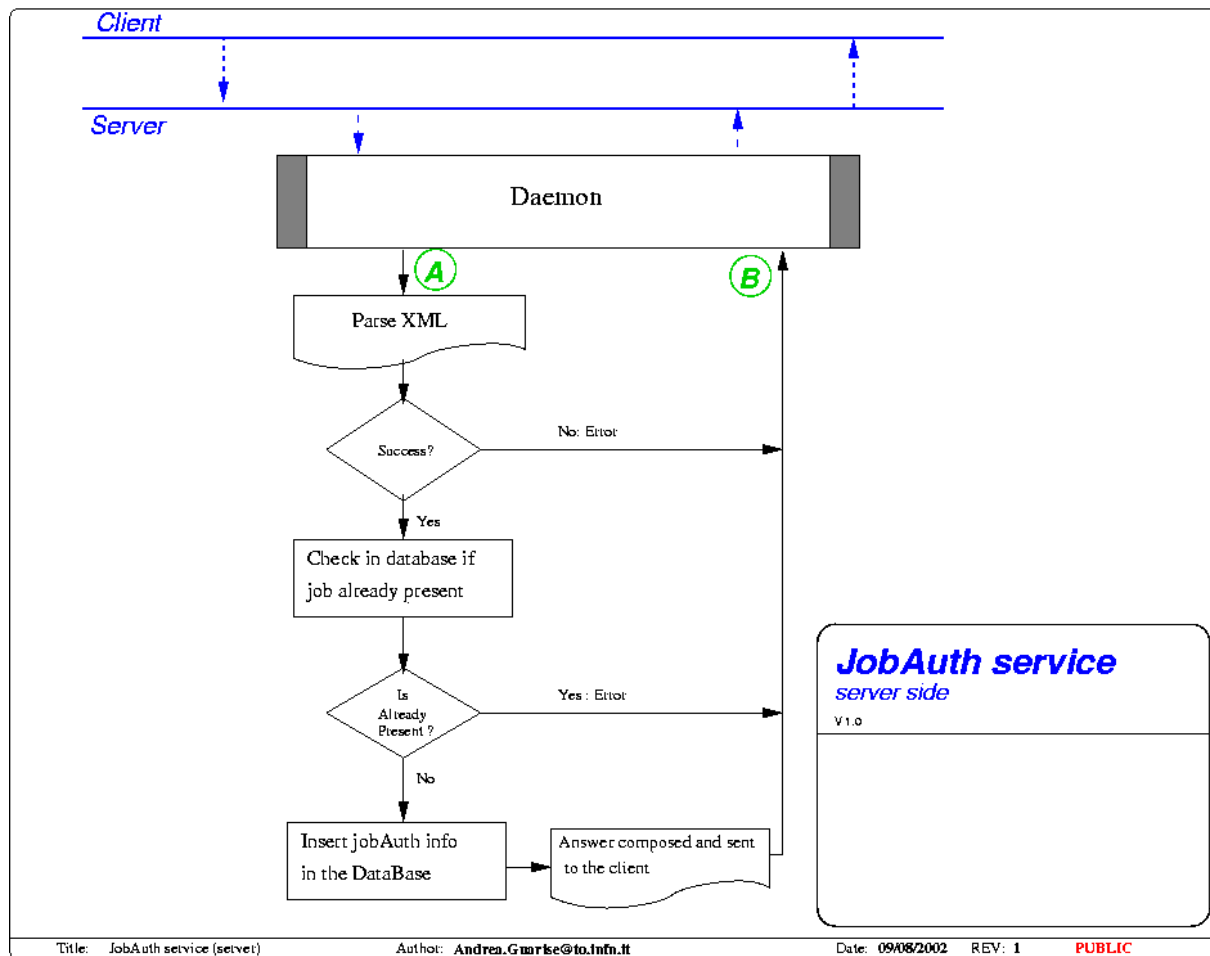


Figure 7: JobAuth server flowchart

XML A, Request from the client.

Description: This message contains the information needed by the JobAuth service. The optional tags identify information that can be retrieved independently, although these parameters are present for debugging purpose.

Message:

```

<HLR type="JobAuth_request">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
  <JOB_AUTH_INFO>
    <DG_JOBID>
      Dg_jobid of the job (MANDATORY)
    </DG_JOBID>
    <SUBMISSION_TIME>
  
```

```
    timestamp of the submission to the WMS in GMT (OPTIONAL)
  </SUBMISSION_TIME>
  <USER_CERT_SUBJECT>
    user X509 certificate subject (OPTIONAL, retrievable via GSI)
  </USER_CERT_SUBJECT>
</JOB_AUTH_INFO>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: This message contains the return status of the job authorization process, 0 on success >0 on failure.

Message:

```
<HLR type="JobAuth_answer">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
  <JOB_AUTH_INFO>
    <DG_JOBID>
      Dg_jobid of the job, it is used also as the transaction ID in the HLR Database
    </DG_JOBID>
    <STATUS>
      return code, 0: success, >0 failure
    </STATUS>
  </JOB_AUTH_INFO>
</BODY>
</HLR>
```

5.4.2. UI - JobAuth service interaction diagram

In Figure 8 we present the interaction diagram between the DataGrid User Interface and the DGAS JobAuth service. The diagram also contains the interaction with the module responsible of ensuring that the HLR furnished by the user is a trusted one. The latter is not really part of the UI – JobAuth service interaction, but is an essential part of the process, so it seems appropriate to show it here.

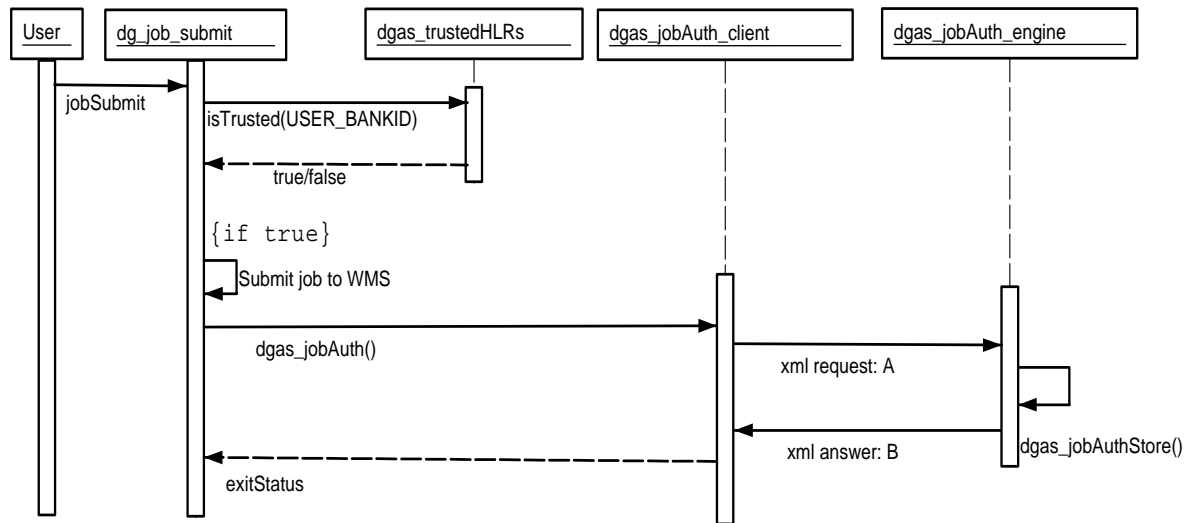


Figure 8: UI - JobAuth service interaction diagram

5.4.3. UI-JobAuth service deployment diagram

The objects needed for this service to work properly should be deployed on the User Interface and on the User HLR server as illustrated in Figure 9

The information passed in the various steps are:

dgas_jobAuth()

Parameter	Description	Type
User_acct_bank_id	Identify the user bank server. The job authorization info will be sent to that bank server.	Mandatory
DG_JobID	Job identifier	Mandatory
User_certSubject	X509 certificate subject. Identifies the user, if the connection towards the server is authenticated via GSI this information can be retrieved directly by the server, so it is optional.	Optional

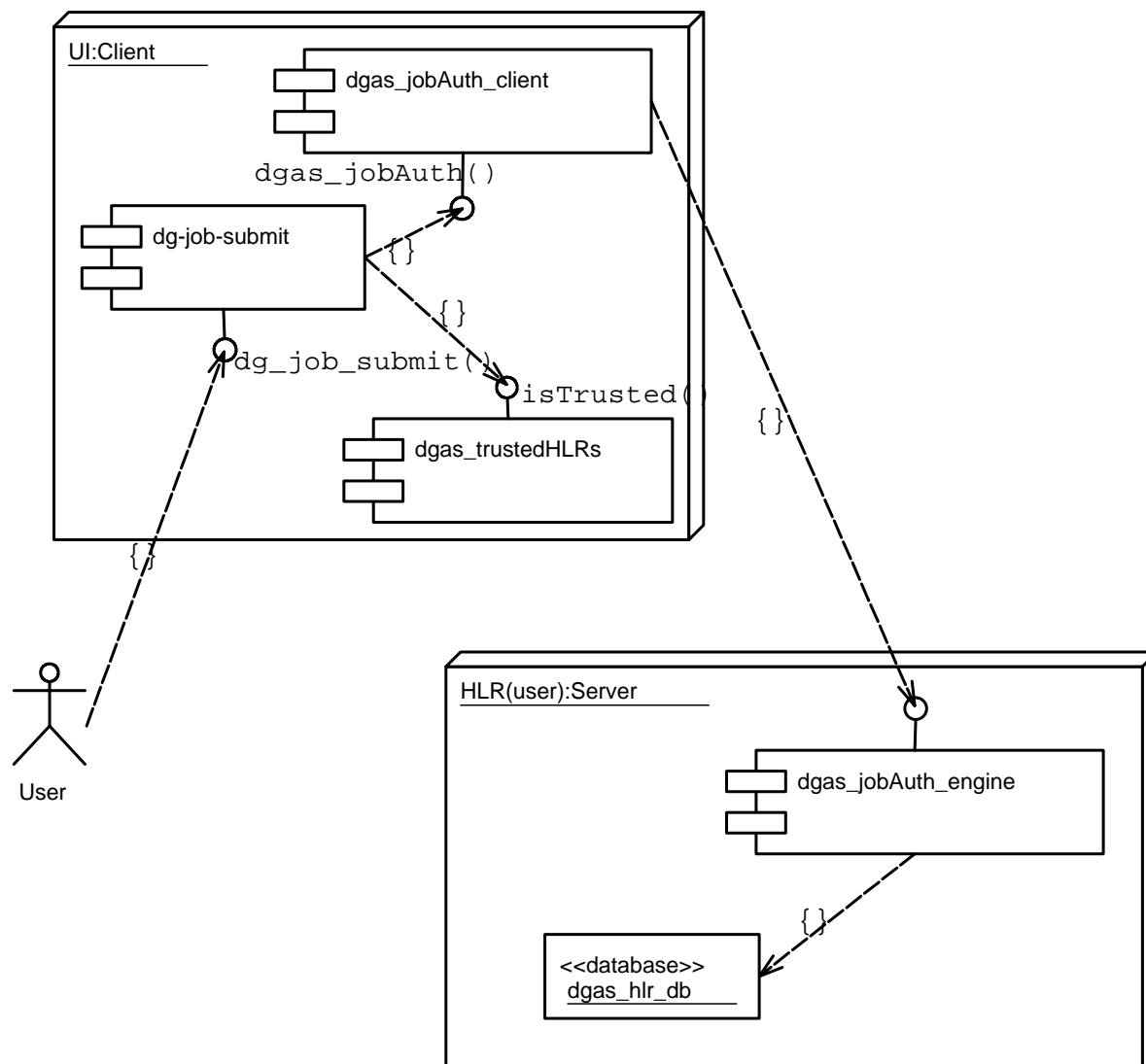


Figure 9: UI - JobAuth service deployment diagram

5.5. RB – HLR USERAUTH SERVICE

One of the key assumptions in an economic ruled computing environment is that beside the usual “static” user authorization mechanisms there is also an economic authorization phase. A user can run jobs on the resources on which he already has an a priori static authorization only if he has enough credits to pay the resource for that job.

This can be seen as a “second stage” authorization mechanism that dynamically extends the authorization rules of the system.

This “fund adequacy” check can be strict or loose. In the first case the User will be authorized only if he has enough credits left to cover the whole job cost, which has to be somehow estimated. In the second case the job will be authorized according to a less tight policy. For example a job can be authorized if the user has a positive balance in his account. If the job cost is greater then the available credits, the user account balance will become negative and no new jobs will be authorized until the debt is absorbed.

In the bootstrap phase of the DataGrid testbed the accounting policy will be even looser than the latter. Job will be always authorized and users will have no debt limit.

So this service is not really important at this moment, but it will be essential if a tighter policy will be adopted by the DataGrid community.

5.5.1. HLR userAuth service flowcharts

In Figure 10 and Figure 11 we present the flowcharts of this service. The service \

used by the Resource Broker to check the user job request against the economic authorization policy. Note that currently a user request will always be authorized in order to help the users gain confidence with the system.

The cost algorithm used in the client module to estimate the job cost is the same as used by the ATM service to compute the final cost for the job after its execution.

It is clear that the algorithm needs an estimation for the CPU_TIME used by the job. If this estimation is not available, a default reference value can be used, like for example, the max cpu_time for the given queue.

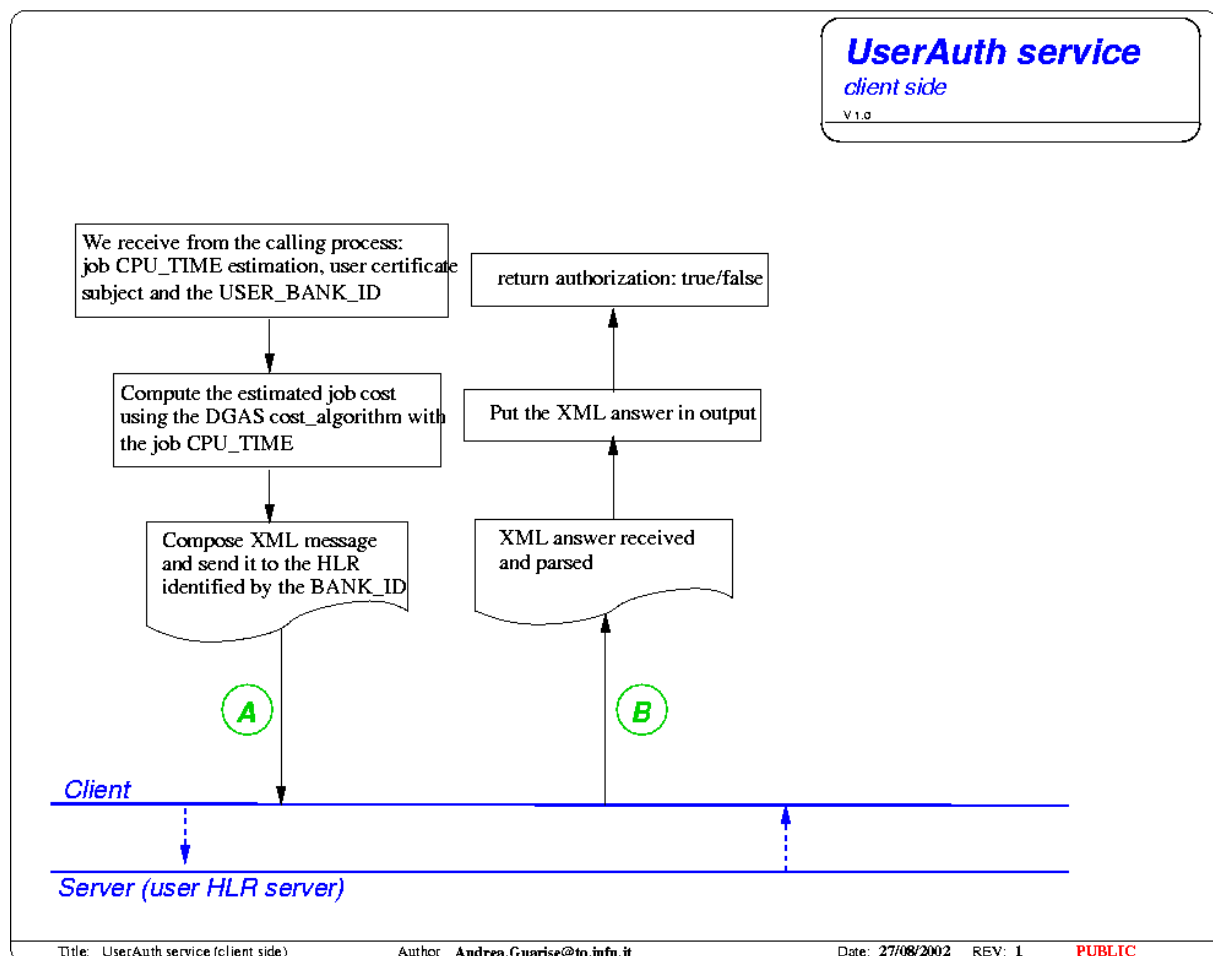


Figure 10: UserAuth service, client side

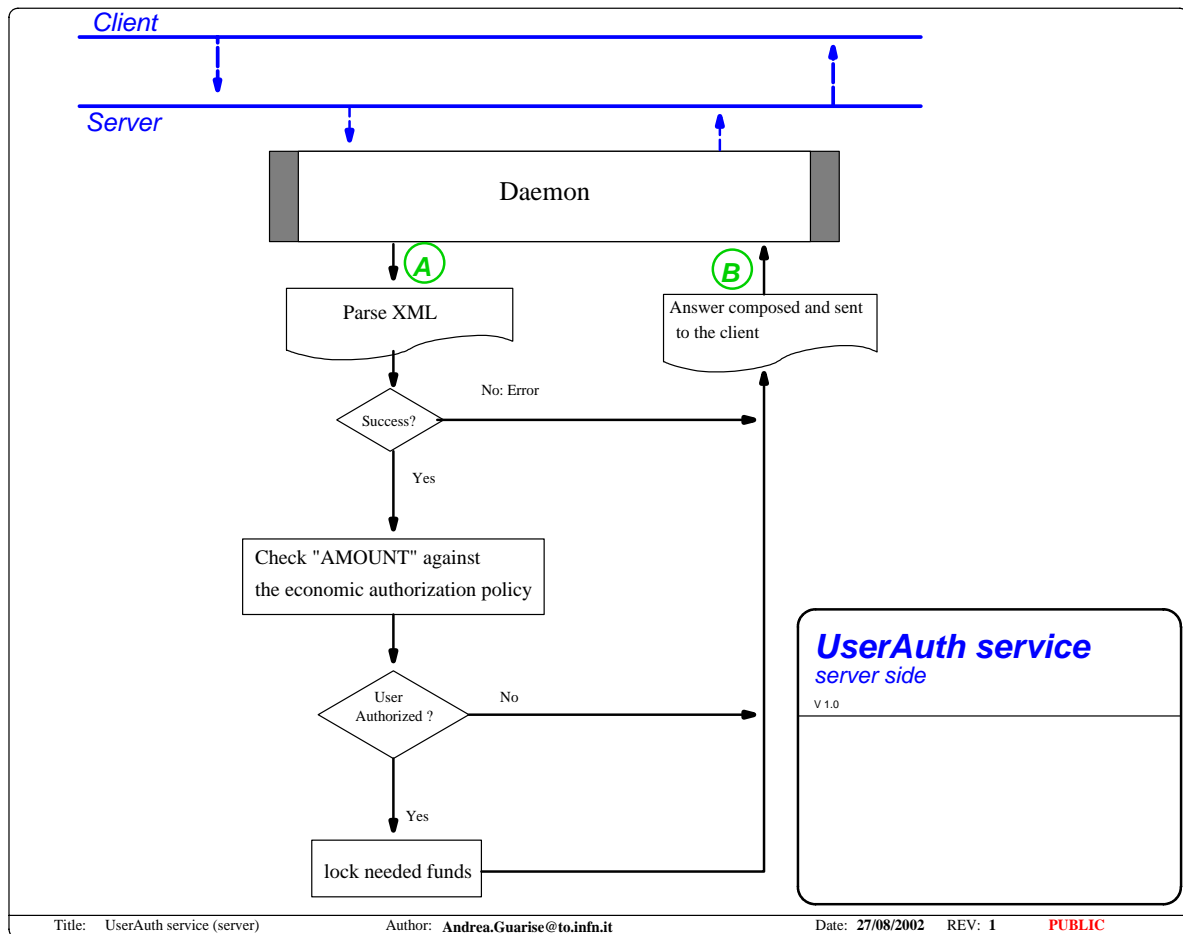


Figure 11: UserAuth service, server side

XML A, Query from the client.

Description: The client, is called by the RB to check that the needed funds are available on the user account.

Message:

```

<HLR type="User_Auth_query">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
<USER_AUTH_INFO>
<USER_CERT_SUBJECT>
User X509 certificate subject; used to identify the user.
</USER_CERT_SUBJECT>
<AMOUNT>
Amount of grid credits needed by the user
</AMOUNT>
</USER_AUTH_INFO>
    
```

```
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: The server verifies the user account status and sends back a positive or negative reply.

Message:

```
<HLR type="User_Auth_answer">
  <HEAD>
    <VER>
      1.0
    </VER>
  </HEAD>
  <BODY>
    <USER_AUTH_INFO>
      <USER_CERT_SUBJECT>
        User x509 certificate subject
      </USER_CERT_SUBJECT>
      <AMOUNT>
        Amount of grid credits needed by the user.
      </AMOUNT>
      <AUTHORIZATION>
        1 = true; 0=false.
      </AUTHORIZATION>
    </USER_AUTH_INFO>
  </BODY>
</HLR>
```

5.5.2. RB - HLR service interaction diagrams

In Figure 12 we present the interaction diagram between the DataGrid Resource Broker and the DGAS HLR user economic authorization service.

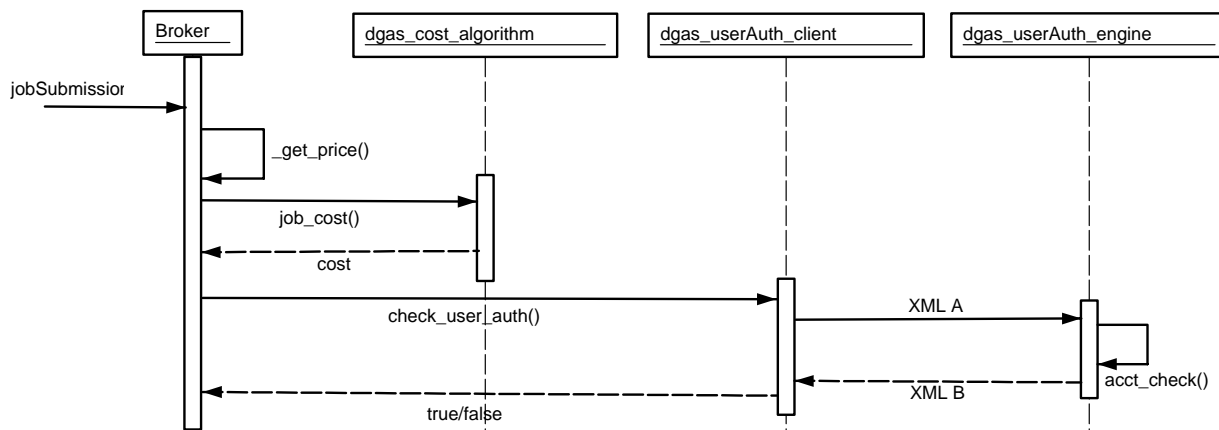


Figure 12: RB-HLR user authorization interaction diagram

When the Broker receives a job submission request, it first chooses the right resource according to both resources characteristics and prices. Then it uses the `job_cost()` algorithm to obtain an estimation of the job cost. This estimation is clearly dependent upon the information given by the user in the JDL.

If the user specified the `CPU_TIME` required for the job, the cost is computed with this value. Otherwise a default value can be used.

Once the Broker obtains the cost estimation from the cost algorithm, it asks the `dgas_userAuth_client` if the user has the economic authorization for such a job and receives a positive or negative answer depending on the user's fund status and on the authorization policy (always true in DataGrid bootstrap phase).

5.5.3. RB-HLR service deployment diagram

The objects needed for this service to work properly should be deployed on the Resource Broker and on the USER HLR server as illustrated in Figure 13.

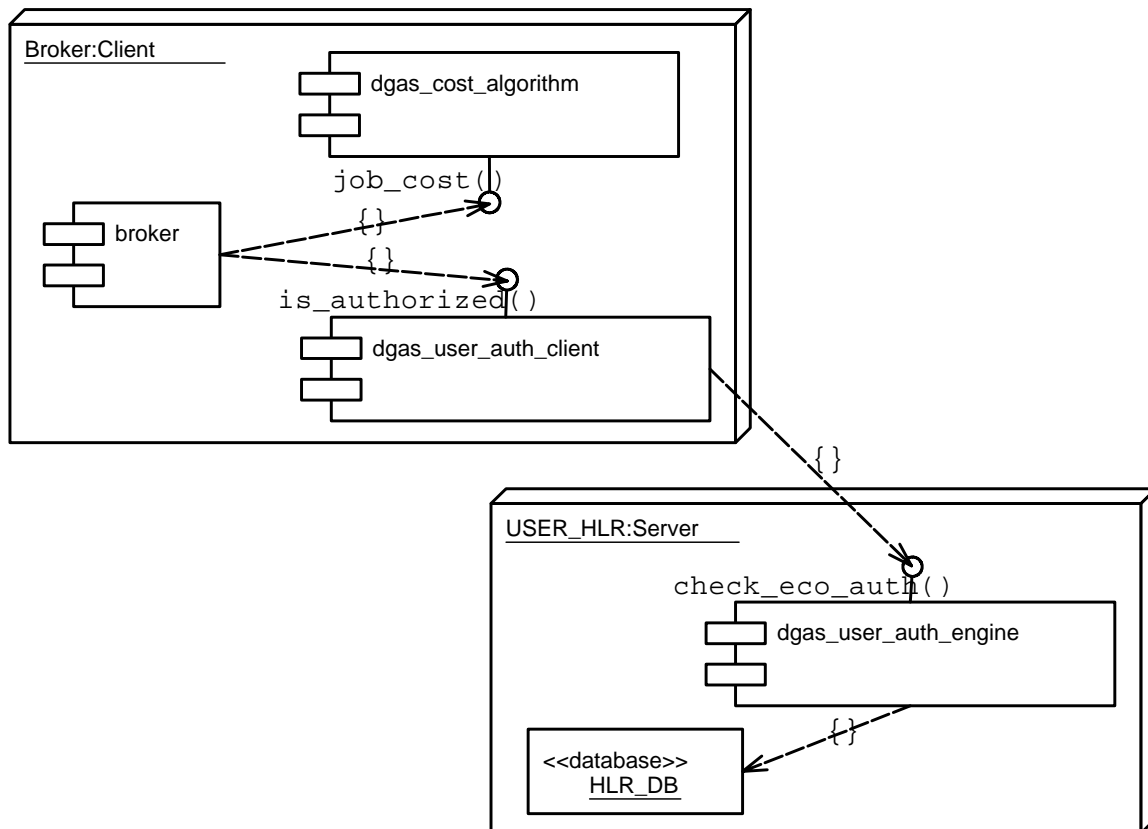


Figure 13: RB-HLR user authorization deployment diagram

The information passed in the various steps are:

Job_cost()



Parameter	Description	Type
ResPrice	Latest known price for the resource.	Mandatory
Usage Records	Used to estimate the cost. For the estimation only CPU_TIME required by the job is needed. If this can't be retrieved in the JDL a default value is set, like for example the maximum CPU_TIME for that queue.	Mandatory

Is_authorized()

Parameter	Description	Type
USER_ACCT_BANK_ID	Grid-univocal identifier for the user bank server; it must be given by the user in the JDL.	Mandatory
USER_X509_SUBJECT	Used to identify the user in the HLR database	Mandatory
COST	Estimated cost for the job.	Mandatory

5.6. PRICE AUTHORITY SERVICE

Figure 14 and Figure 15 represent the Price Authority client and server respectively. This service furnishes a valid price quotation when needed.

When the resource broker needs a price quotation, that value is returned along with the GMT time of interrogation and that price (considered an offer) is valid for the following interval (default is one hour) during which the job can be submitted. If the job is not submitted within this interval, submission is refused and the user is notified. When a RB requests a new price calculation the result is stored in the SQL database with its generation time. This value remains the valid price until the next update, when its expiration becomes the new generation time.

In this way a job assignment by the RB will be subject to the original price even though the price should change before the job is executed. This concept is equivalent to the principle that the price on an invoice should be the same as that on the corresponding order based on an unexpired offer. The required server is found by consulting a table of Pricing Authority addresses associated with the resource HLR involved. The use of GMT guarantees that RB job assignments in one time zone that are honoured in another one will be consistent in time.

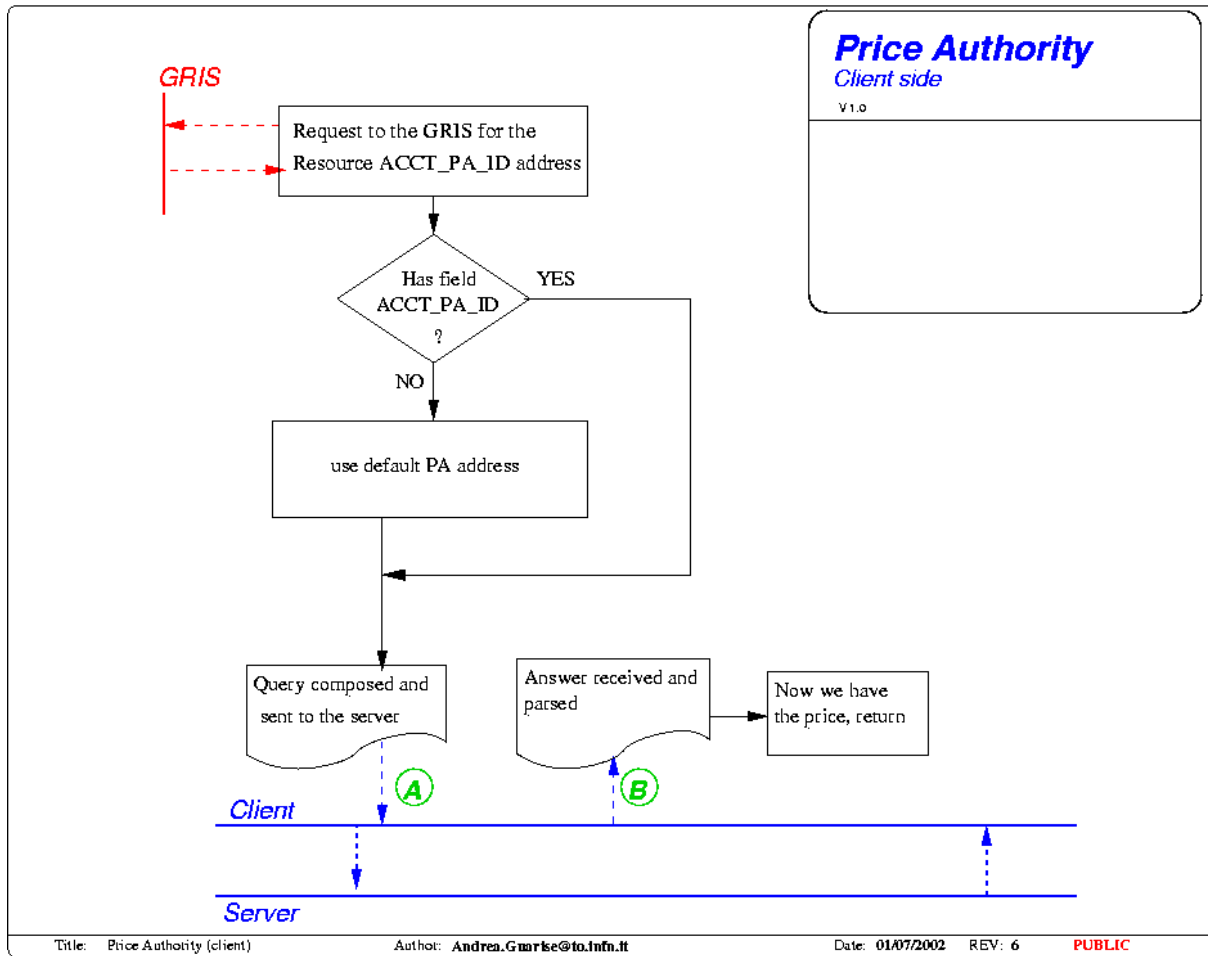


Figure 14 : Price Authority, client side.

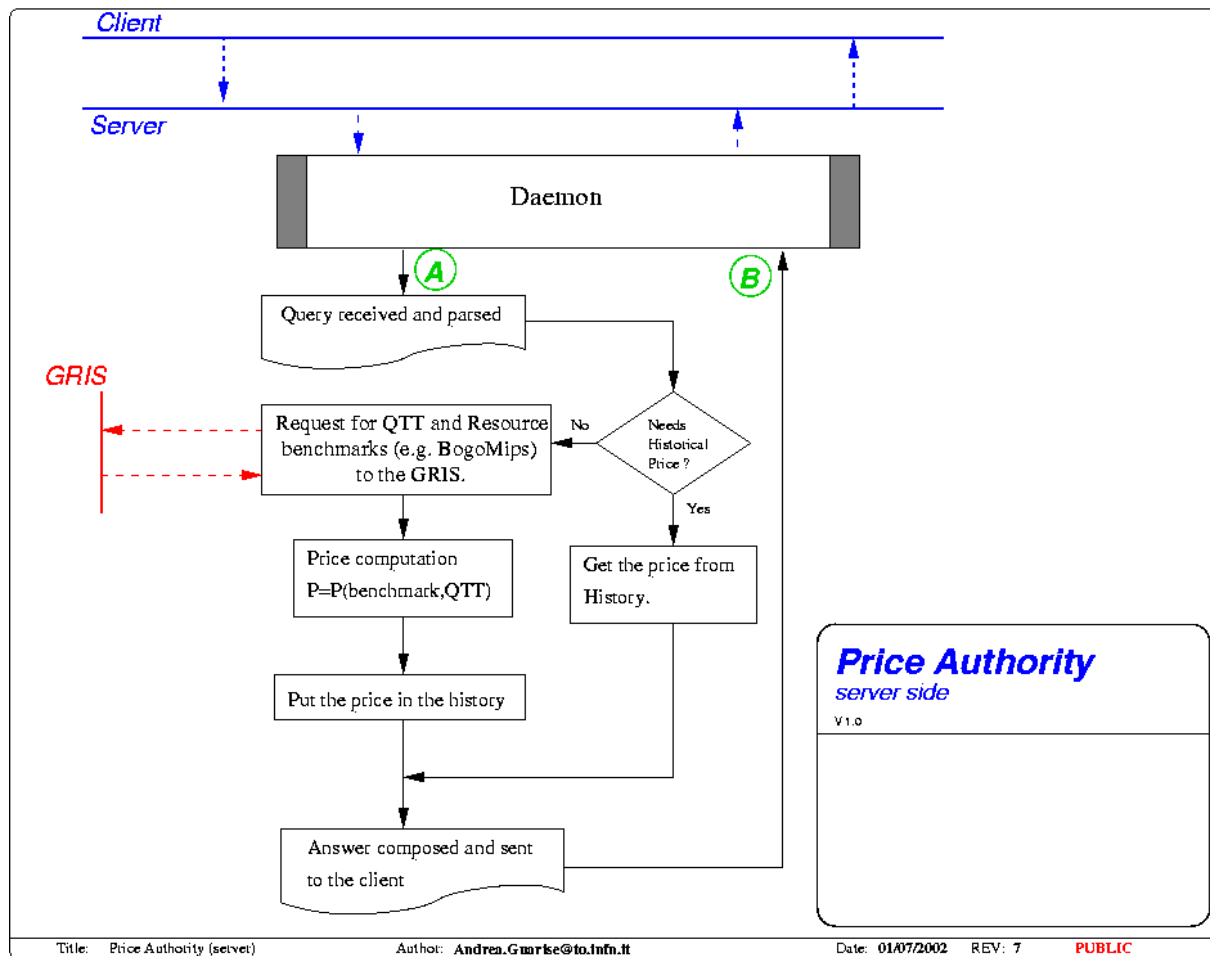


Figure 15: Price Authority, server side.

XML A, Query from the client.

Description: The client, that can be a Sensor, the RB or another HLR asks the price of a given resource at the time expressed by the timestamp in the query.

Message:

```

<HLR type="PA_query">
  <HEAD>
    <VER>
      1.0
    </VER>
  </HEAD>
  <BODY>
    <PRICE_INFO caller="type_of_caller">
<!-- type of caller can be:
      Sensor
      HLR
      RB
-->
  </BODY>
</HLR>
  
```



```
<RES_ID>
  Grid resource ID e.g. the resource GRAM identifier
</RES_ID>
<TIME>
  timestamp (in GMT) for the request
</TIME>
</PRICE_INFO>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: The server retrieves the price (or prices if there are many) of the resource, or computes it if necessary, and sends it to the client.

Message:

```
<HLR type="PA_answer">
  <HEAD>
    <VER>
      1.0
    </VER>
  </HEAD>
  <BODY>
    <PRICE_INFO>
      <RES_ID>
        Grid resource ID
      </RES_ID>
      <TIME>
        needed timestamp (GMT)
      </TIME>
      <PRICE type="cpu || mem || ...">
        resource price (Grid Credits)
      </PRICE>
      <MIN_TTL>
        Minimum validity time for this price (i.e. this price will be valid at
        Least until TIME + MIN_TTL)
      </MIN_TTL>
    </PRICE_INFO>
  </BODY>
</HLR>
```

If, for some reason, the server is unable to retrieve the resource price, the body of the message will contain an entry like:

```
<PRICE_INFO type="error">
  <RES_ID>
    Grid resource Identifier
  </RES_ID>
```

```

<ERROR>
  ErrorCode
</ERROR>
</PRICE_INFO>

```

5.6.1. RB - PA service interaction diagrams

In Figure 16 we present the interaction diagram between the DataGrid Resource Broker and the DGAS Price Authority service.

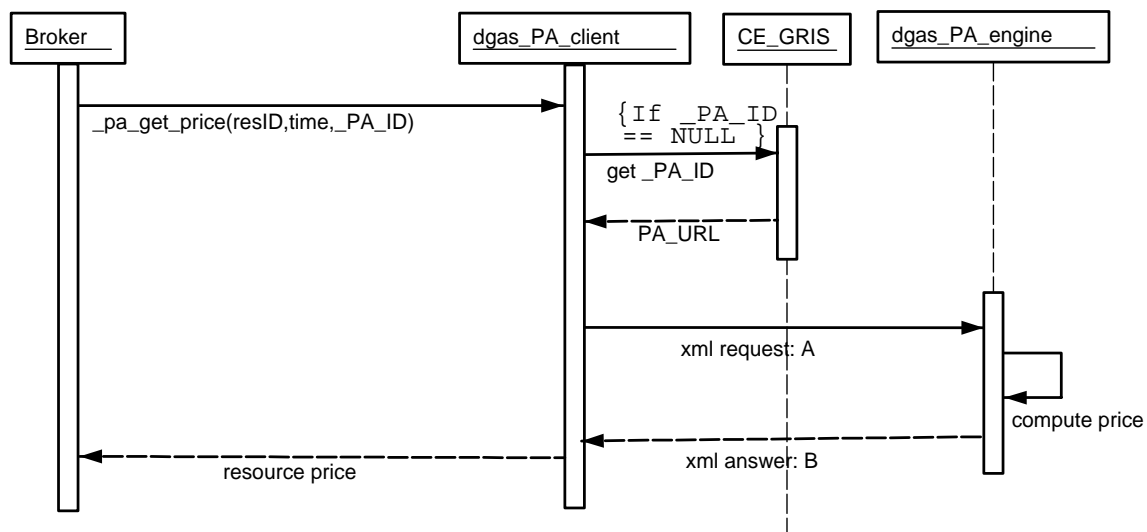


Figure 16: RB - PA service interaction diagram

5.6.2. RB-PA service deployment diagram

The objects needed for this service to work properly should be deployed on the Resource Broker and on the Price Authority server as illustrated in Figure 17

The information passed in the various steps are:

Dgas_pa_get_price()

Parameter	Description	Type
ResID	Grid-univocal identifier for a resource, e.g. the GRAM contact string.	Mandatory
Time	GMT timestamp for the price (the current timestamp if the client is a Resource Broker)	Mandatory
ACCT_PA_ID	Used to contact the right PA for the resource, it should be retrieved by the Broker itself in the resource GRIS.	Optional

Resource price

Parameter	Description	Type
ResID	Grid-univocal identifier for a resource, e.g. the GRAM contact string.	Mandatory
Price	GMT timestamp for the price (the current timestamp if the client is a Resource Broker)	Mandatory
MinimumTTL	Validity time for the resource price.	Mandatory

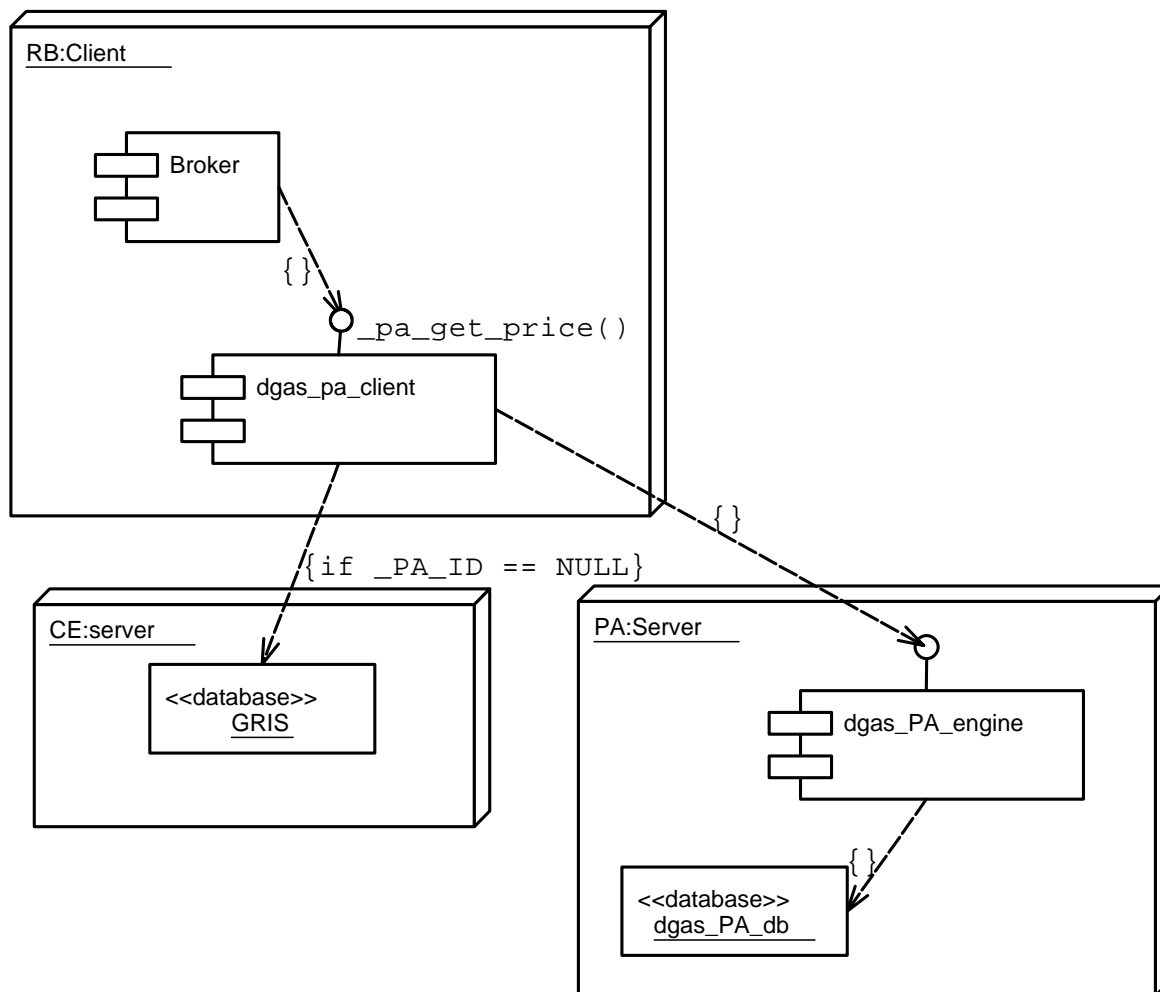


Figure 17: RB - PA service deployment diagram

5.7. USER-HLR UI SERVICE

This “service” represents the User Interface to the HLR service that allows to retrieve information about the accounts and is not part of the job submission flow.

Figure 18 represents the client side of the user-HLR interface. This side is activated by using a shell application which translates the user request into a properly formatted XML query and sends it to the appropriate server. When a response is received from the server (Figure 19) it is parsed first of all for whether the corresponding query was accepted as being authorized⁴. If not, the ensuing negative response is presented to the user. Else the results of the query are visualized and the client process exits.

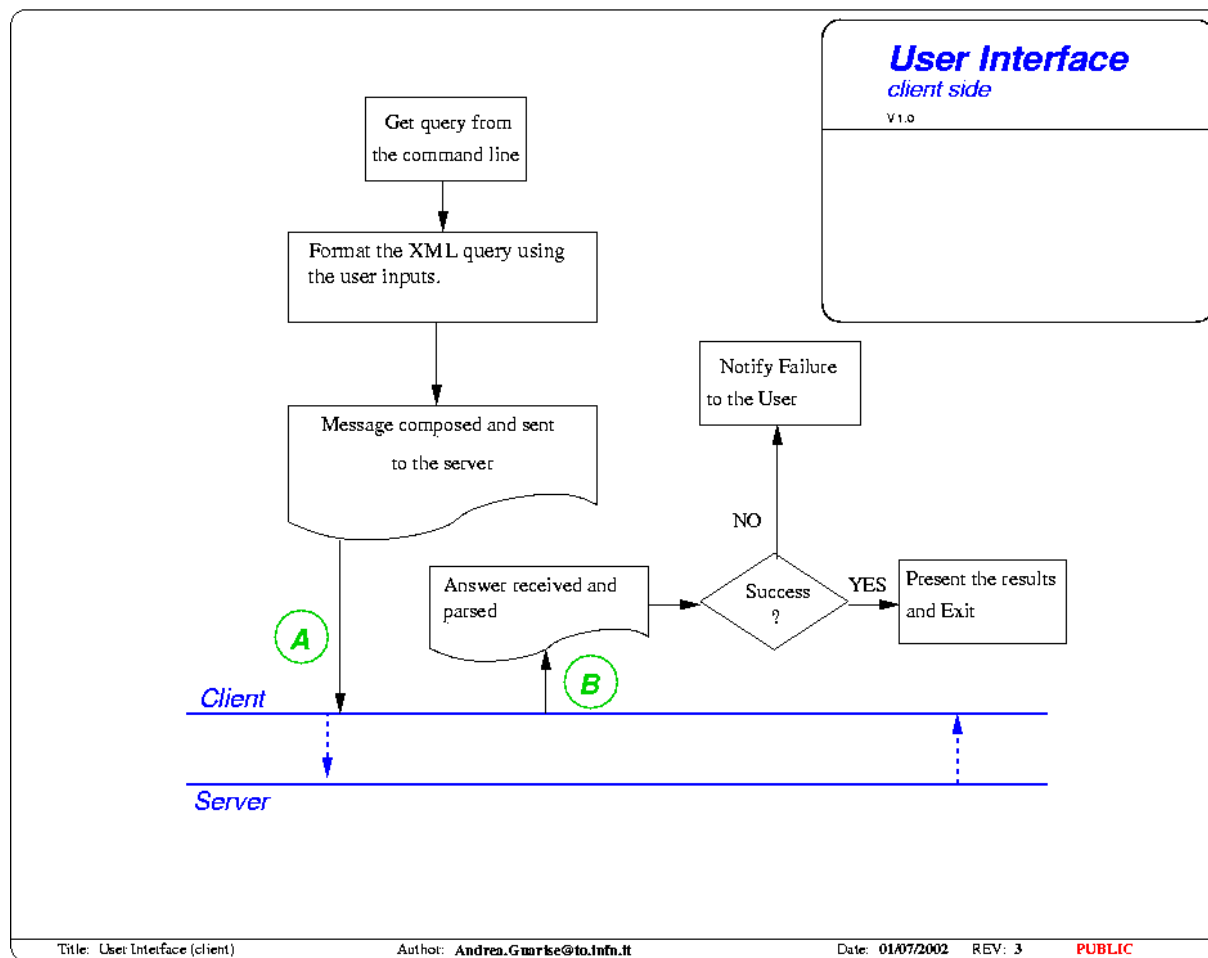


Figure 18: User Interface, client side.

⁴ In the current version of the software this authorization mechanism is still not implemented.

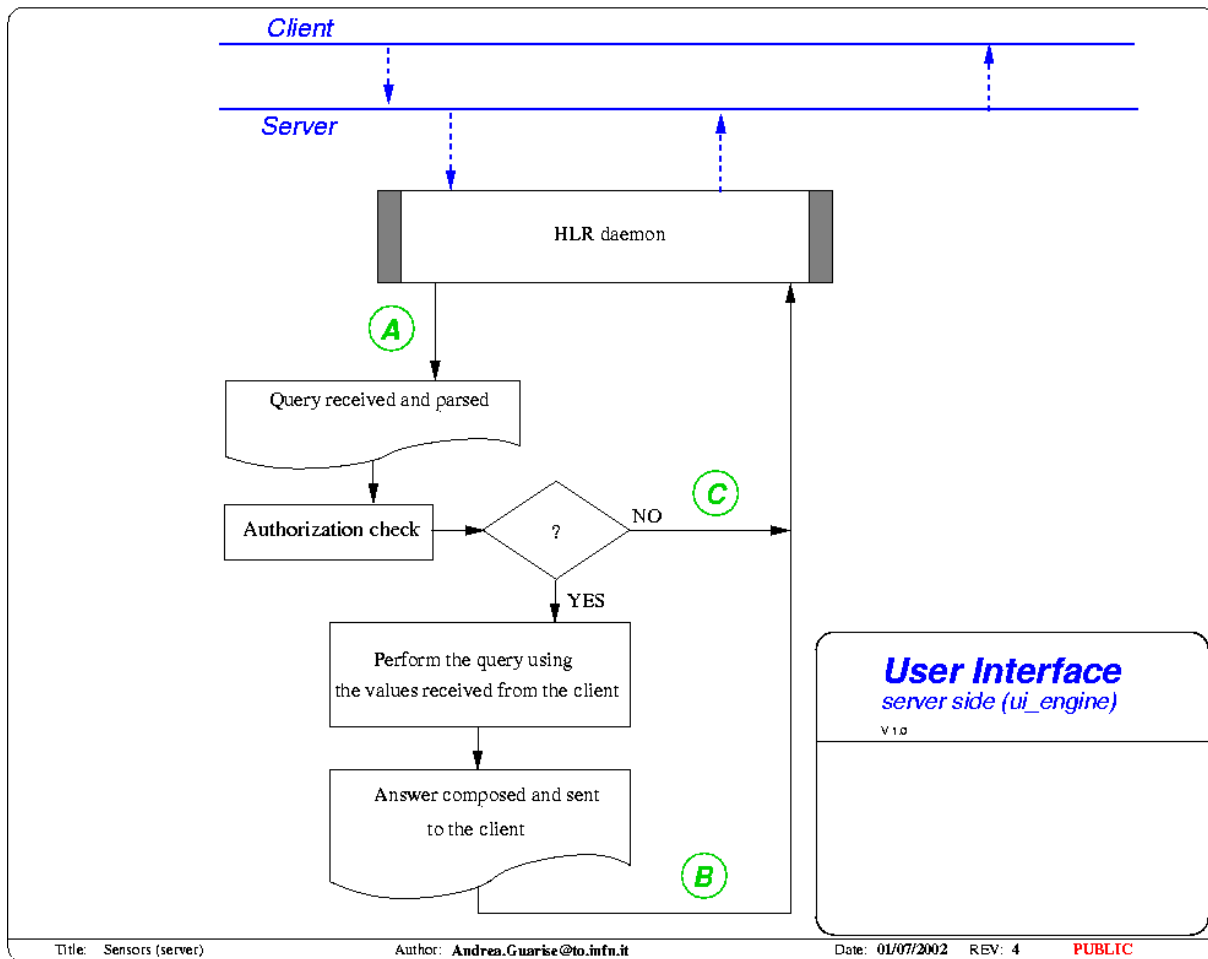


Figure 19: User Interface, server side.

XML A, Query from the client.

Description: The client asks the available information about a user that can be identified via his e-mail, subject of the X509 certificate or by his group (in this case there may be a multiple answer).

Message:

```

<HLR type="UI_query">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
  <USER_INFO>
    <EMAIL>
      u_info.email
    </EMAIL>
    <CERT_H>
      u_info.cert_subject
    </CERT_H>
  </USER_INFO>
</BODY>
</HLR>
  
```



```
<GID>
  u_info.gid
</GID>
</USER_INFO>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: The server retrieves the needed information and if the requesting user is authorized to manage those information it composes the answer message.

Message:

```
<HLR type="UI_reply">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
  <USER_INFO>
    <HLR_UID>
      u_info.uid
    </HLR_UID>
    <EMAIL>
      u_info.email
    </EMAIL>
    <DESC>
      u_info.decr
    </DESC>
    <CERT_H>
      u_info.cert_subject
    </CERT_H>
    <GID>
      u_info.gid
    </GID>
    <FUND_ASS>
      u_info.total
    </FUND_ASS>
    <FUND_BKD>
      u_info.booked
    </FUND_BKD>
    <FUND_SPT>
      u_info.spent
    </FUND_SPT>
  </USER_INFO>
</BODY>
</HLR>
```

5.8. SENSORS – ATM SERVICE

It should be clear that the accounting system needs much information about the user's job in order to correctly determine its cost. The cost of a job is computed from its resource usage, so it is important there be on every Computing Element some sensors that can report to the accounting system the resource usage for each job.

Many types of sensors are needed, one for each computing resource that we want to consider, for example: CPU time, RAM allocation, Disk Usage etc...

These sensors must, however, share a common set of characteristics and functionalities that we explain hereafter.

First, it should be clear that the job being monitored must be unambiguously identified by its DGJobId. Similar conditions hold for the Grid User that owns the job and for the Grid Resource that executes it. Both the resource and the user should be uniquely identified, for example the user by his X509 certificate subject or the resource by its contact string.

Every sensor must be a lightweight application in order not to interfere excessively with the computation while returning the values of the information monitored.

We can now explain how we intend to use these sensors in the accounting environment.

When the sensor system on the CE gathers information about a job⁵ it sends the following information to the user HLR (See Figure 20), some parameters are mandatory, other are optional and used only during debugging and testing:

User information:

Parameter	Type
USER_ACCT_BANK_ID	Mandatory

This information is used to identify the user HLR server on which the user has the bank account he wants to debit for the current job. Since in principle a user may have accounts on more than one HLR this value must be specified by the user manually when submitting a job, maybe via a mandatory parameter in the JDL. This info can then be propagated to the CE in an ENV variable.

Job information:

Parameter	Type
DGJobID	Mandatory
Job Submission Timestamp	Optional**
RES_ACCT_PA_ID	Optional *
RES_ACCT_BANK_ID	Optional *
User X509 certificate Subject	Optional**

⁵ For the first implementation of the system we agreed with WP4 that the information will be collected and sent to the DGAS software only after the job completion, but if future needs will require it, this process can happen also periodically during the job run.

Resource GRID ID	Mandatory
------------------	-----------

**RES_ACCT_PA_ID* and *RES_ACCT_BANK_ID* will be inserted in the CE GRIS in version 2.0 of the EDG software, so it will be feasible to retrieve these two parameters directly from the GRIS, but until this happens they must be specified manually.

** The Job submission timestamp and the User cert subject can be retrieved directly by the accounting system.

Usage records:

Parameter	type
CPU_TIME	Mandatory
WALL_TIME	Optional

Then on the user HLR the job cost can be computed and the invoice amount transferred to the resource HLR.

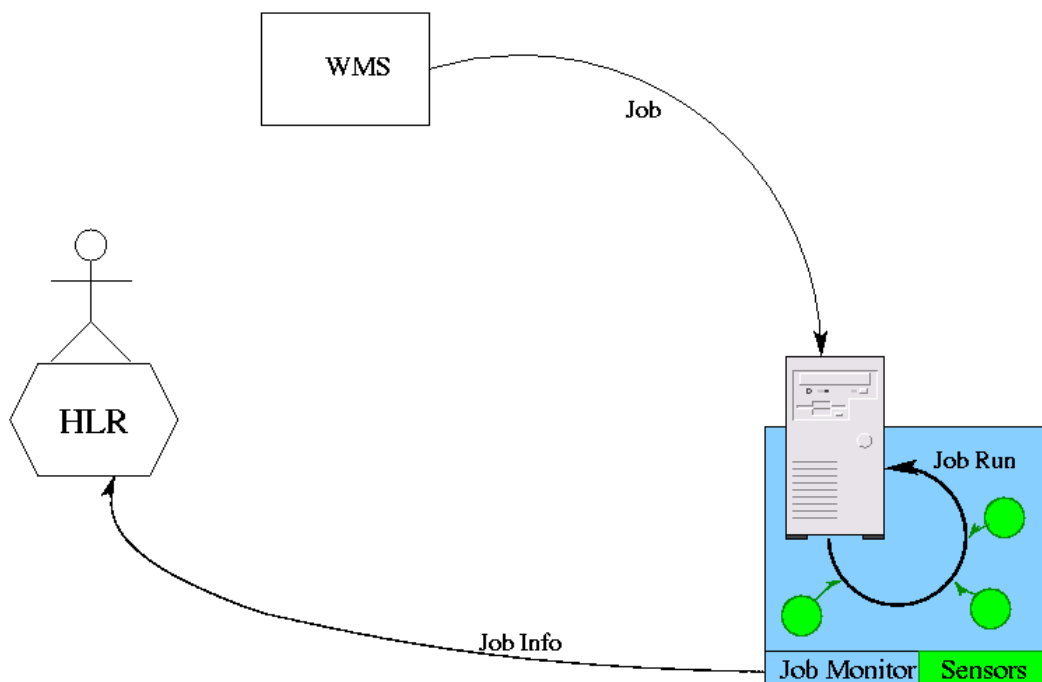


Figure 20 JobMonitor and sensors scheme

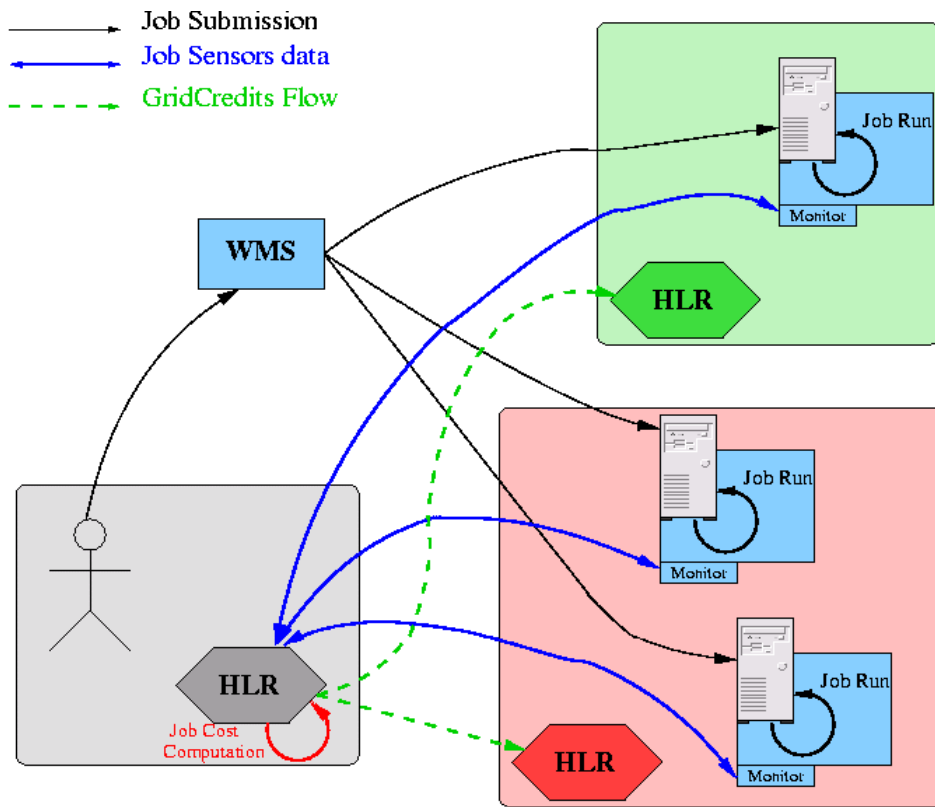


Figure 21 Job Monitor and cost computation data flow

Figure 21 illustrates the intended flexibility of the submission system through which a job may be partitioned and submitted to separate CEs or submitted initially to one CE and, after being checkpointed, submitted to a different CE.

5.8.1. ATM Service flowcharts

Figure 22 and Figure 23 represent the Computing Element Sensors client and server, respectively. This service is activated when a job terminates. The client combines the job static information with the consumption measurements furnished by the executing CE as input to the server that calculates the total cost. The price used is the value valid at the time of job submission. For this reason all job submission timestamps must be given in GMT in order to facilitate a World Wide Grid.

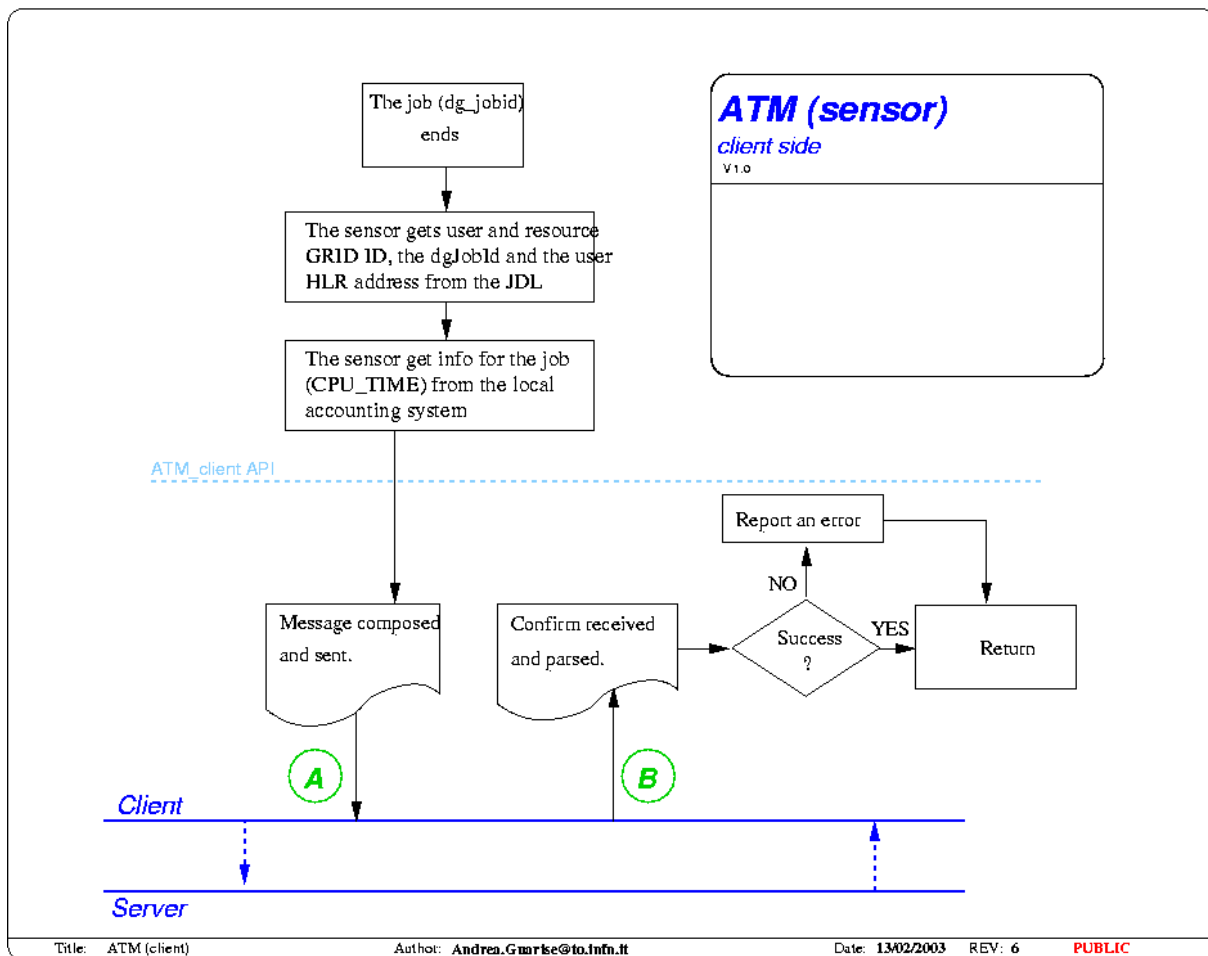


Figure 22: ATM (Job payment), client side.

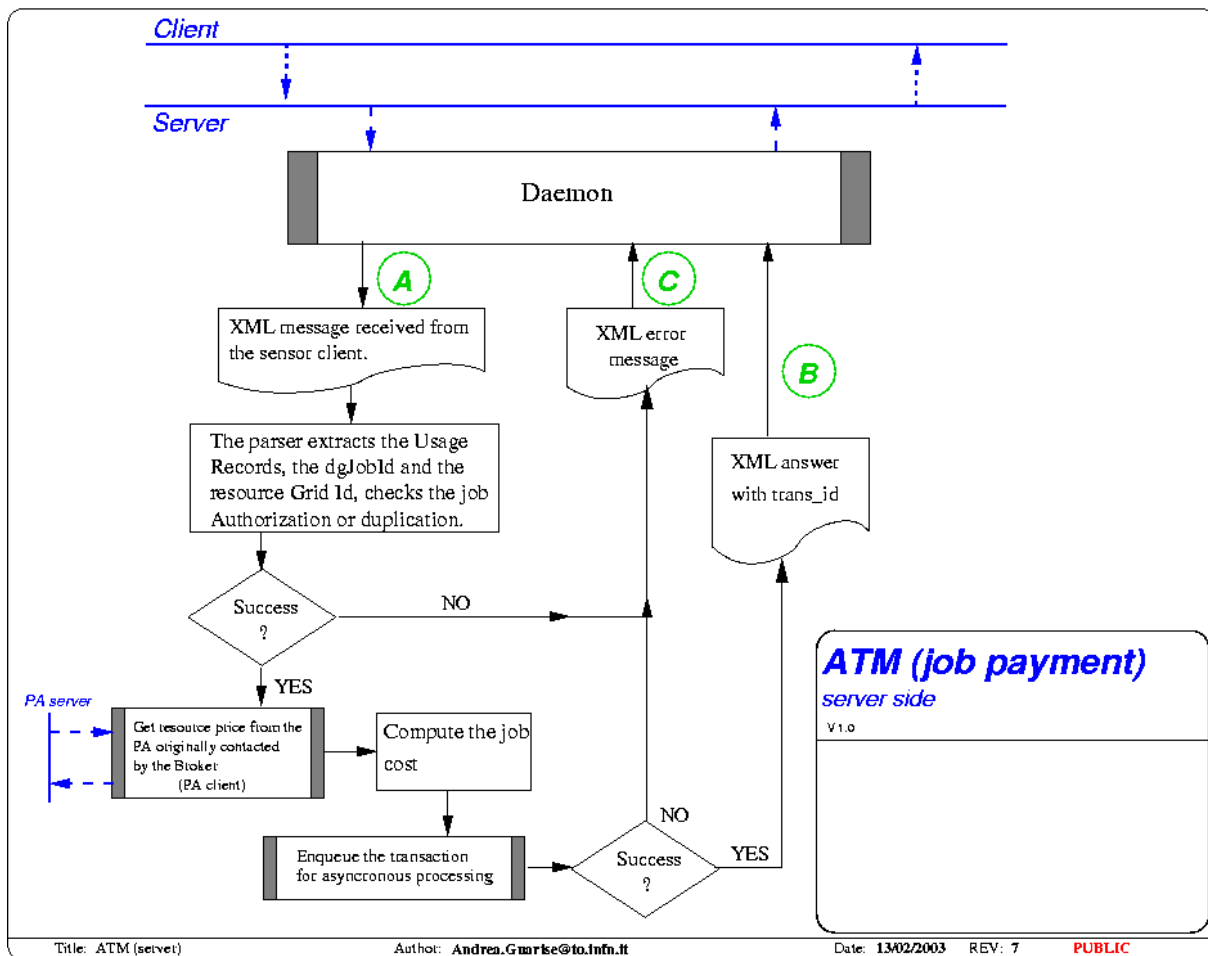


Figure 23: ATM (Job payment), server side.

XML A, Request from the client.

Description: This message contains the information needed by the ATM server to compute the job cost, debit the user and credit the resource. The PA_URL field is needed when requesting an historic price, since it is important to ask the same PA that previously did the assignment.

Message:

```

<HLR type="ATM_request">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
<JOB_PAYMENT>
<DG_JOBID>
Dg_jobid of the job (MANDATORY)
</DG_JOBID>
<SUBMISSION_TIME>
    
```

```
    timestamp of the submission (OPTIONAL since available elsewhere)
</SUBMISSION_TIME>
<RES_ACCT_PA_ID>
    hostname:port:X509_certsobject of the Price Authority of the resource.
    It is used to contact the server and perform mutual authentication.
    (OPTIONAL)
</RES_ACCT_PA_ID>
<RES_ACCT_BANK_ID>
    hostname:port:X509_certsobject of the HLR that contains the resource account.
    It is used to contact the server and perform mutual authentication.
    (OPTIONAL)
</RES_ACCT_BANK_ID>
<USER_CERT_SUBJECT>
    user X509 certificate subject (OPTIONAL)
</USER_CERT_SUBJECT>
<RES_CERT_SUBJECT>
    resource X509 certificate subject (MANDATORY)
</RES_CERT_SUBJECT>
<JOB_INFO>
    <CPU_TIME>
        CPU time consumed by the job
    </CPU_TIME>
    <WALL_TIME>
        Wallclock time used by the job
    </WALL_TIME>
</JOB_INFO>
</JOB_PAYMENT>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: This message summarizes the transaction processed by the ATM service and tells the user if the transaction concluded successfully or not. This message can also be considered as a receipt for the transaction. (In future versions a system for ensuring the authenticity of a receipt will be provided)

Message:

```
<HLR type="ATM_answer">
<HEAD>
<VER>
1.0
</VER>
</HEAD>
<BODY>
    <JOB_PAYMENT>
        <DG_JOBID>
```

```
Dg_jobid of the job, it is used also as the transaction ID in the HLR Database
</DG_JOBID>
<SUBMISSION_TIME>
  timestamp of the submission
</SUBMISSION_TIME>
<RES_ACCT_PA_ID>
  PA that furnished the resource price
</RES_ACCT_PA_ID>
<RES_ACCT_BANK_ID>
  HLR of the resource that was credited for the job.
</RES_ACCT_BANK_ID>
<USER_CERT_SUBJECT>
  user X509 certificate subject
</USER_CERT_SUBJECT>
<RES_CERT_SUBJECT>
  resource X509 certificate subject
</RES_CERT_SUBJECT>
<PAYMENT_INFO>
  <COST>
    Cost of the job
  </COST>
  <STATUS>
    OK | FAILED
  </STATUS>
  <CODE>
    exit code
  </CODE>
</PAYMENT_INFO>
</JOB_PAYMENT>
</BODY>
</HLR>
```

5.8.2. Sensor –ATM_service Interaction diagram

In Figure 24 we describe the interaction diagram between the CE sensor and the ATM-engine on the DGAS User HLR server.

The job information is stored in a queue and then processed asynchronously later. In this way the sensor system needs only to wait the confirmation that the data were correctly received by the ATM service.

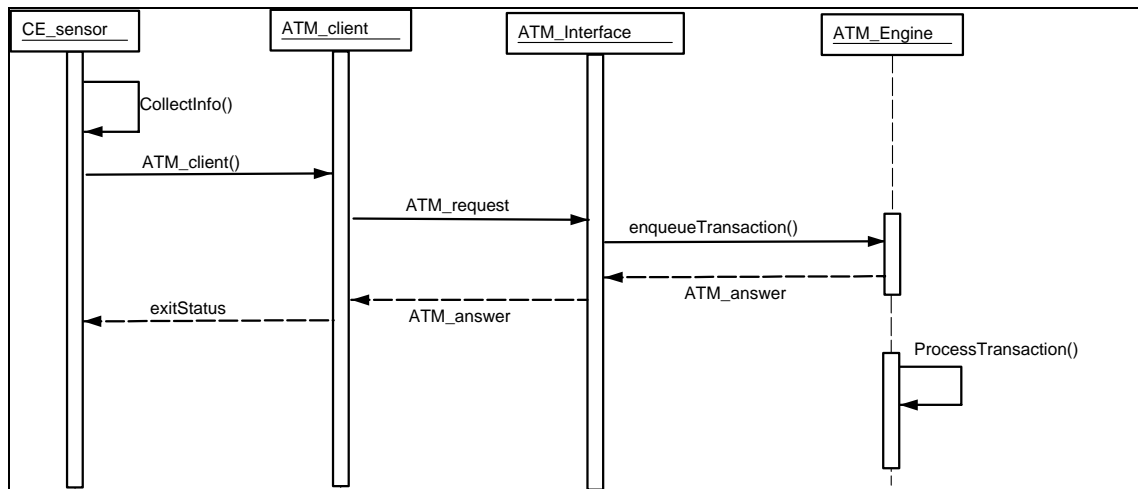


Figure 24: Sensor - ATM engine interaction diagram

5.8.3. Sensor –ATM_service deployment diagram

Figure 25 represents the deployment structure of the relevant sensor and accounting components. Both the current and the future architecture are illustrated.

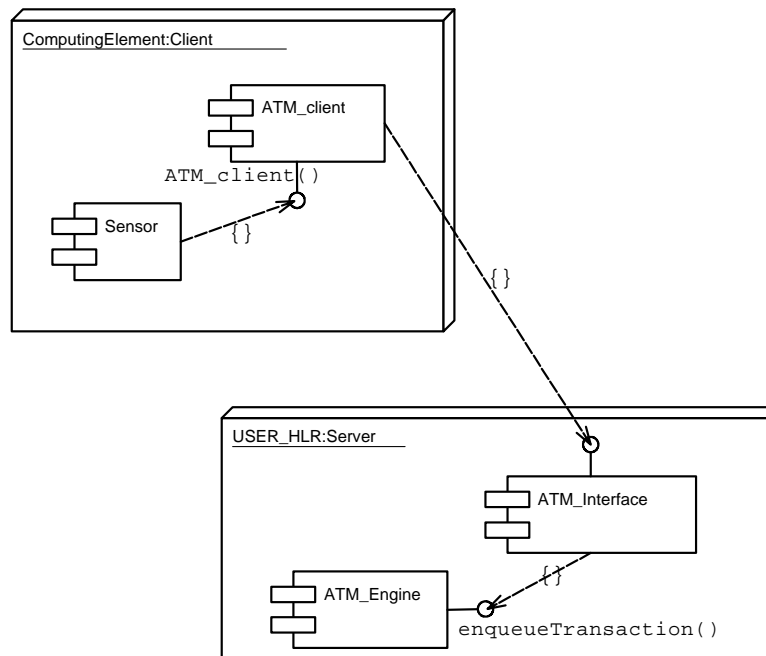


Figure 25: Deployment diagram v2

5.9. GENERAL SCHEME

In this section we describe the working flow of the DGAS software while managing the accounting process for a generic user job.

5.9.1. Job debiting & crediting calls scheme

The process starts when the user job finishes and the monitoring system sends the appropriate information to the user HLR using the ATM service⁶. The server side of the ATM service that runs on the User HLR needs to know the valid price of the resource at the job submission time, thus it uses an instance of the PA client to ask that price to the PA server of the resource (whose address should be available on the resource GRIS). The next step is to compute the job cost and initiating the crediting-debiting process. This is performed by the bank-service client that calls the remote resource HLR bank-service. The resource HLR address should come with the job info or better be available on the resource GRIS. The schema of the above process is shown in Figure 26.

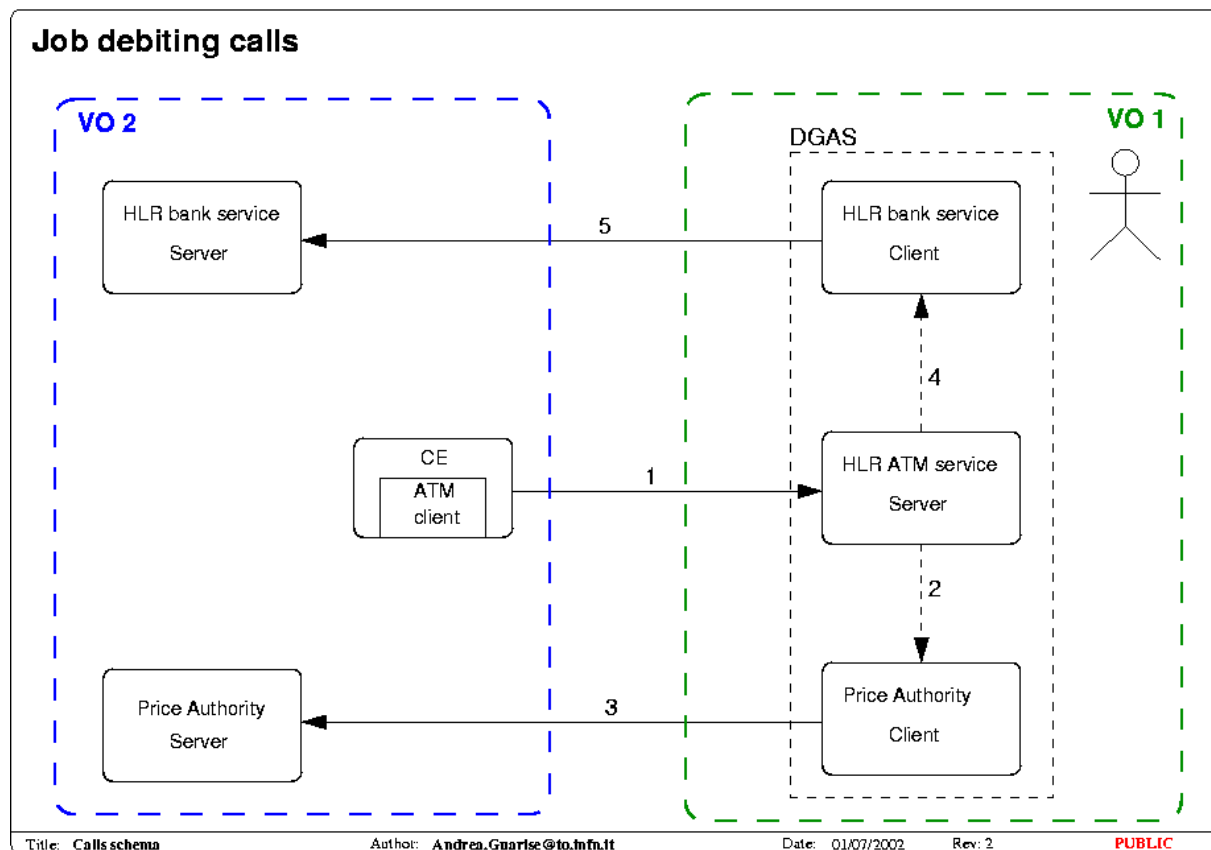


Figure 26: job crediting & debiting calls

⁶ First of all the ATM server checks in the user HLR DB if the dgJobId of the current job is registered as valid in order to avoid the processing of transactions originating from fake requests. This implies that the job must be registered by the user in the HLR DB immediately after it's submission. The best way to achieve this is to implement this functionality directly in the *UserInterface* that can perform this operation on behalf of the user with his credentials.

Assuming the existence of one DGAS per VO it is clear that the internal organization of the DGAS service on the two VOs (the user and the resource one in the example) should be symmetrical as shown in Figure 27, where both the VOs have their own users and resources.

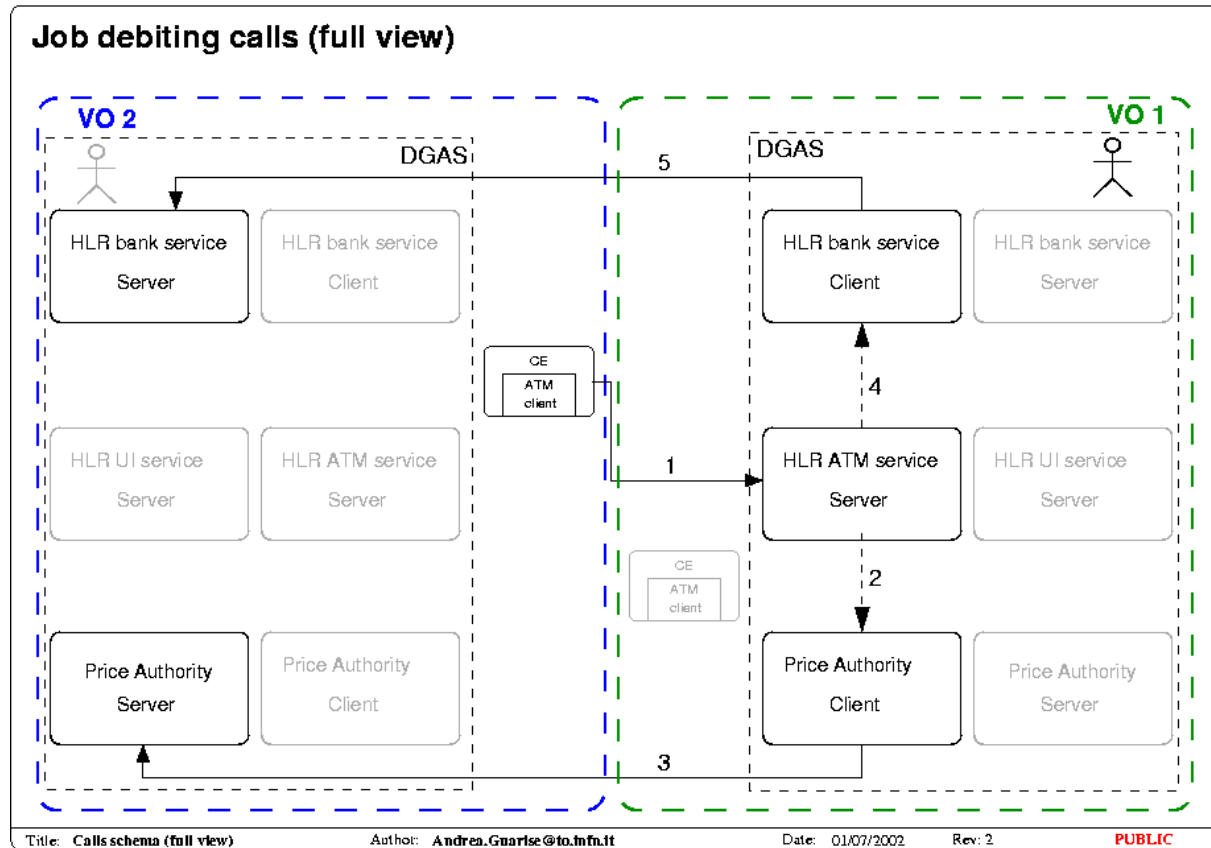


Figure 27: DGAS service components

5.9.2. Job debiting & crediting main flow

Here we describe the flowchart of the whole process of job debiting and crediting that is illustrated in Figure 28. In the figure dashed arrows mean internal (to the process) calls, while continuous ones mean external XML network calls. The contents of those calls is illustrated hereafter.

1: ATM Client → ATM server

Type: XML (see 5.8.1)

- *DgJobId*: Used to uniquely identify the job that generates the transaction.
- *Submission_timestamp*: Used to identify the valid price of the resource.
- *RES_ACCT_PA_ID*: In the form: host:port:X509_subject. Used to retrieve the URL of the PA for the resource that runs the job. If the GSI security is enabled for the PA this information is also needed for retrieving the expected contact string used during the mutual authentication between client and server.

- *RES_ACCT_BANK_ID*: In the form: host:port:X509_subject. Used to retrieve the URL of the HLR for the resource that runs the job. If the GSI security is enabled for the HLR this information is also needed for retrieving the expected contact string.
- *User_X509_subject*: Subject of the user X509 certificate, used to unambiguously identify the Grid User in the DGAS environment.
- *Res_Grid_ID*: Resource Grid identificative, used to unambiguously identify a Grid Resource in the DGAS environment.
- *Usage record*: Usage info used to compute the job cost.⁷
 - i. *CPU_TIME*
 - ii. *WALL_time*
 - iii. ...

2: GRIS → PA client

Type: LDAP query

- *RES_ACCT_PA_ID*: Used to retrieve the PA url, this query is necessary only if the *RES_ACCT_PA_ID* isn't already furnished by the ATM client in step 1.

3: PA client → PA server

Type: XML (see 5.4)

- *Res_Grid_ID*: Used to identify the resource on the PA.
- *Price_timestamp*: The timestamp of the job submission time. The price returned will be the one valid at that moment.

4: PA server → PA client

Type: XML (see 5.4)

- *Res_Grid_ID*: Used by the PA to identify the resource.
- *Price*: Price of the resource at job submission time.
- *Time*: Begin time of the validity period for this price.

5: GRIS → PA server⁸

Type: LDAP query.

- *EstimatedTraversalTime*
- *Max N. of available CPUs*
- *N. of free CPUs*
- *MaxRunningJobs*
- *CurrentRunningJobs*

⁷ These usage records are exemplificative only. A working group within the GGF is currently working on the definition of the set of record needed to correctly describe a job in a general manner.

⁸ The information needed by the Price Authority to compute the resource prices is still mainly to be defined.

- *CurrentPendingJobs*

6: PA client → Cost algorithm

Type: Internal

- *Price*: Valid resource price for the job.
- *Usage Info*: Held information about the job resource usage.
 - *Cpu_time*: Cpu time used by the job.

7: Cost algorithm → HLR bank_service client

Type: Internal

- *JobCost*: Cost of the given job.

8: GRIS → HLR bank service client

Type: LDAP query

- *RES_ACCT_BANK_ID*: Used to retrieve the URL of the resource HLR. Used if not already specified in the info sent by the ATM_client in step 1.

9: HLR bank service client → HLR bank_service server

Type: XML (see 5.3)

- *Res_Grid_ID*: Identifies the resource being credited.
- *User_X509_subject*: Identifies the user being debited.
- *DgJobId*: Identifies the job.
- *JobCost*: Cost of the given job.
- *ConnectionInfo*: Contains information about the current connection, for security reasons.
 - *RemoteHostName*: The Host Name of the client.
 - *ContactString*: GSI contact string of the caller, used in the Authentication phase.

10: HLR bank service server → HLR bank_service client

Type: XML (see 5.3)

- *Resource_X509_subject*: Identifies the resource being credited.
- *User_X509_subject*: Identifies the user being debited.
- *DgJobId*: Identifies the job.
- *JobCost*: Cost of the given job.

11: Debit user call

Type: internal

- *Resource_X509_subject*: Identifies the resource being credited.
- *User_X509_subject*: Identifies the user being debited.
- *DgJobId*: Identifies the job.

- *JobCost*: Cost of the given job.

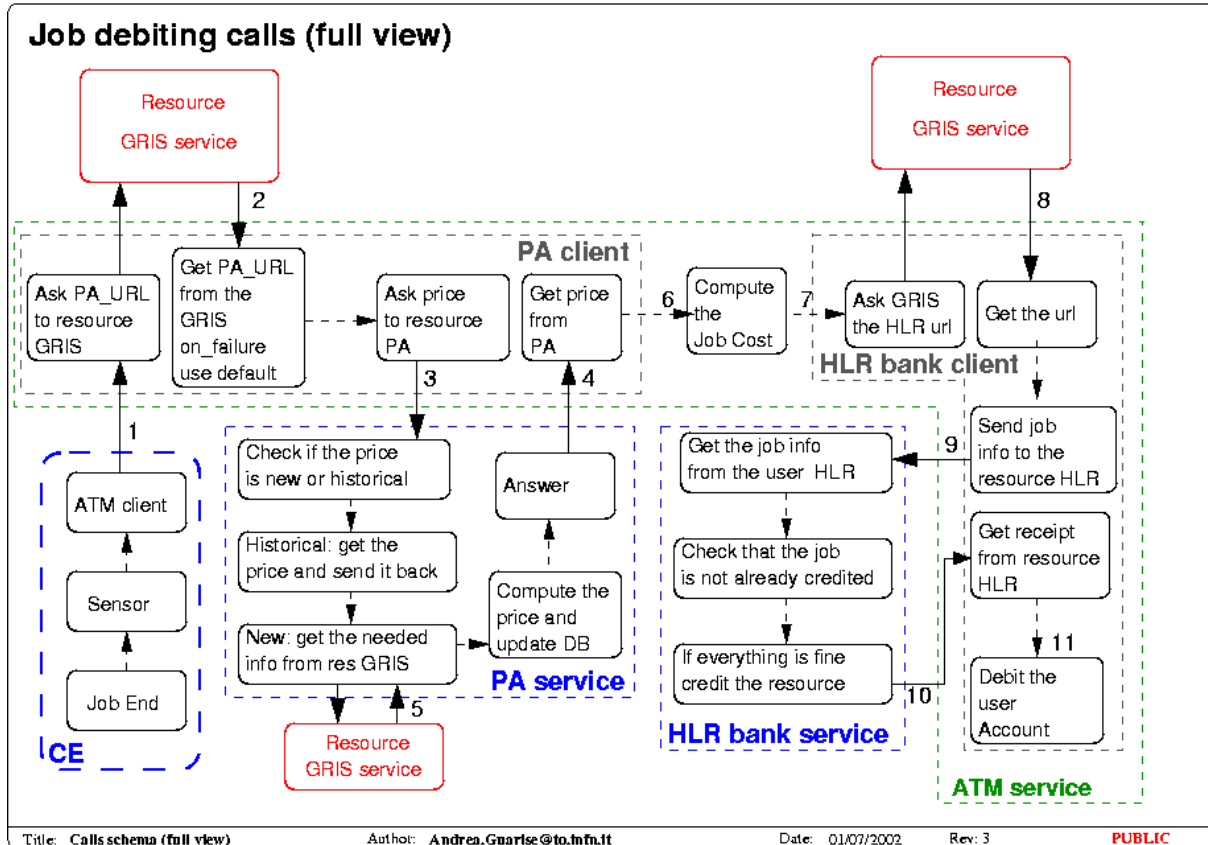


Figure 28 Crediting & debiting flow

6. DATABASES DESCRIPTION

The DGAS software is based upon two relational databases:

1. The Bank Service Database.
2. The Pricing Authority Database.

Here we provide a brief description of the logical organization of the databases, with the simple intent to explain what types of information are held in the database. The implementative organization of these tables may be different.

6.1. BANK SERVICE DATABASE

The bank service database stores the economic information about users, resources and jobs⁹.

User_info – stored information about the users		
Uid	string	User internal id
Email	string	User email
Descr	string	User brief description
Cert_subject	string	Subject of the X509 user's certificate
Gid [→ group_info::gid]	string	Id of the group to which the user belongs
Assigned	int	Amount of credits assigned to the user
Reserved	int	Amount of credits currently reserved by the user for the pending jobs.
Spent	int	Amount of credits already spent by the user jobs.

group_info – stored information about groups of users		
Gid	string	Group internal id
Descr	string	Group brief description
Fid [→ fund_info::fid]	string	Id of the fund to which the group belongs
Assigned	int	Amount of credits assigned to the whole group
Reserved	int	Amount of credits currently

⁹ In the tables, the items in **Bold** typeface identify the fields used in the API as public identifiers for the information in the table. I.e. each query of the databases performed via the APIs should use those keys in the request.



		reserved by the group for the pending jobs.
Spent	int	Amount of funds already spent by the users jobs.

Fund_info – stored information about the funds		
Fid	string	Fund internal id
Descr	string	Fund brief description
Assigned	int	Amount of credits assigned to the fund.
Spent	int	Amount of credits already spent.

resource_info – stored information about the resources		
Rid	String	Resource internal id
Descr	String	Resource brief description
Cert_subject	string	Subject of the X509 resource’s certificate.
Gid [→ group_info::gid]	string	Group to which the resource belongs.
Total	int	Amount of credits earned by the resource.

tr_info – stored information about the economic transactions processed by the HLR		
Transaction_id	String	transaction internal id, if the transaction originates from a job, it must be equal to the dg-jobid of the job.
from_cert_subject	string	Subject of the X509 certificate of the account that is being debited.
to_cert_subject	string	Subject of the X509 certificate of the account that is being credited.
from_hlr_url	string	URL of the HLR that sends the payment.
to_hlr_url	string	URL of the HLR that receives the payment.
Amount	int	Amount of grid-credits transferred in the transaction.
timestamp	int	Unix timestamp of the transaction (GMT)

6.2. PRICING AUTHORITY DATABASE

The Pricing Authority Database holds the information about the prices assigned to every resource by the corresponding PA.

Prices have a well defined minimum time to live and a well defined time of assignment. An history of the prices already assigned to a resource is kept. Thus in every moment it is possible to retrieve the past price of a resource.

resource_info – stored general pricing information about the resources		
Cert_subject	string	Subject of the X509 resource's certificate.
Price_ttl	int	Validity duration (in seconds) (default and minimum =3600).

resource_price – stored historical prices for a resource (one table per resource)		
Cert_subject	string	Subject of the X509 resource's certificate.
Time	Int	Begin of the price validity period. N.B. The price is considered valid until a new price generation is requested.
Price	Int	Price for the resource (In future version there will be a vector of prices, one for every accountable feature.)

The generation of this database should be automatic. Every time that someone requests the price for a resource, if the resource isn't already known to the PA, a new entry is inserted in **resource_info** and a new **resource_price** table is generated. Additionally, if the validity of the actual price has expired, a new price is calculated when the RB requests a price quotation.



7. CONCLUSIONS

We have designed and developed a series of services which we think can furnish a reliable accounting function for the DataGrid WP1 middleware. Throughout its realization, we have tried to remain independent of economic models so that any reasonable economic model can be integrated when chosen.

For the present all economic values are in GridCredits and no conversion of GridCredits to other monetary units is considered.



8. ANNEXES

8.1. SOFTWARE DESCRIPTION

The DGAS software is being mainly developed in C++, the name and version of the software needed to compile the software is listed below, with the rpm packages where available.

Software name	Version	Description	RPM package
gcc	2.96	Compiler	gcc-2.96-98 gcc-c++-2.96-98
libstdc++	2.96	C++ Standard Libraries	libstdc++-2.96-98 libstdc++-devel-2.96-98
glibc	2.2.4	System standard libraries	glibc-2.2.4-19.3 glibc-devel-2.2.4-19.3 glibc-common-2.2.4-19.3
mysql	3.23.41	RDBMS	mysql-3.23.41-1 mysql-devel-3.23.41-1 mysql-server-3.23.41-1

The software has been developed under Linux RedHat 7.2, other versions known to work are:
RedHat 7.3

If you know of other version/distributions working please let us know so that we can update this list.