



DataGrid

DATA GRID ACCOUNTING SYSTEM PUBLIC INTERFACES ARCHITECTURE - V 0.5



Document identifier: **DataGrid-01-TED-0131-0_5**

Date: **27/08/2002**

Work package: **WP1**

Partner: **INFN**

Document status **DRAFT**

Deliverable identifier:

Abstract: In this document we describe in detail the architecture of the public interfaces to the DGAS accounting system. This version of the document describes the V1.0 of DGAS, distributed within the V2.0 of the EDG software.

Delivery Slip

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

Document Log

Issue	Date	Comment	Author
0.1	31/07/2002	First public version (sensors only)	A.Guarise,
0.5		Complete document	A.Guarise, A.Werbrouck

Document Change Record

Issue	Item	Reason for Change

Files

Software Products	User files
Word	DataGrid-01-TED-0131-0_0-acctinterfaces
Acrobat	DataGrid-01-TED-0131-0_0-acctinterfaces.pdf

CONTENT

1. INTRODUCTION	4
1.1. OBJECTIVE OF THIS DOCUMENT	4
1.1.1. <i>Status of this document</i>	4
1.2. APPLICATION AREA	4
1.3. TERMINOLOGY	4
2. PREMISES	6
3. UI – JOBAUTH SERVICE	7
3.1. SERVICE ARCHITECTURE	7
3.1.1. <i>JobAuth service flowcharts</i>	7
3.1.2. <i>UI - JobAuth service interaction diagrams</i>	9
3.1.3. <i>UI-JobAuth service deployment diagram</i>	10
3.2. SERVICE API	11
3.2.1. C++	11
4. RB – PA SERVICE	13
4.1. SERVICE ARCHITECTURE	13
4.1.1. <i>PA service flowcharts</i>	13
4.1.2. <i>RB - PA service interaction diagrams</i>	15
4.1.3. <i>RB-PA service deployment diagram</i>	16
4.2. SERVICE API	17
4.2.1. C++	17
5. RB – HLR USERAUTH SERVICE	19
5.1. SERVICE ARCHITECTURE	19
5.1.1. <i>HLR userauth service flowcharts</i>	19
5.1.2. <i>RB - HLR service interaction diagrams</i>	22
5.1.3. <i>RB-HLR service deployment diagram</i>	23
5.2. SERVICE API	25
5.2.1. C++	25
6. SENSORS – ATM SERVICE	26
6.1. SERVICE ARCHITECTURE	28
6.1.1. <i>ATM Service flowcharts</i>	28
6.1.2. <i>Sensor –ATM_service Interaction diagram</i>	32
6.1.3. <i>Sensor –ATM_service deployment diagram</i>	33
6.2. SERVICE API	34
6.2.1. C++	35
7. ANNEXES	37
7.1. SOFTWARE DESCRIPTION	37

1. INTRODUCTION

1.1. OBJECTIVE OF THIS DOCUMENT

The objective is to provide a detailed description of the interfaces and relationships with other software components external to the DGAS infrastructure but still within the scope of the DataGrid middleware, such as the sensor system, the Resource Broker or the UI.

For further information on the Accounting project see [R1]. For Detailed information on DGAS internals see [R2].

1.1.1. Status of this document

In the following table you can find information about the software revision to which the document refers.

Document revision	Date	Software version	EDG release
1.0		1.0	2.0

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
- [R2] A.Guarise, A.Werbrouck – *DataGrid Accounting System – Architecture –v 0.8* – DataGrid-01-TED-0126-0_8 – http://www.to.infn.it/grid/accounting/curr_arch.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 verify 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. Time spent in a queue before beginning execution.

GMT

Greenwich Mean Time

2. 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 contains 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.

3. 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 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.

3.1. SERVICE ARCHITECTURE

3.1.1. JobAuth service flowcharts

In Figure 1 and Figure 2 we present the flowcharts of this service.

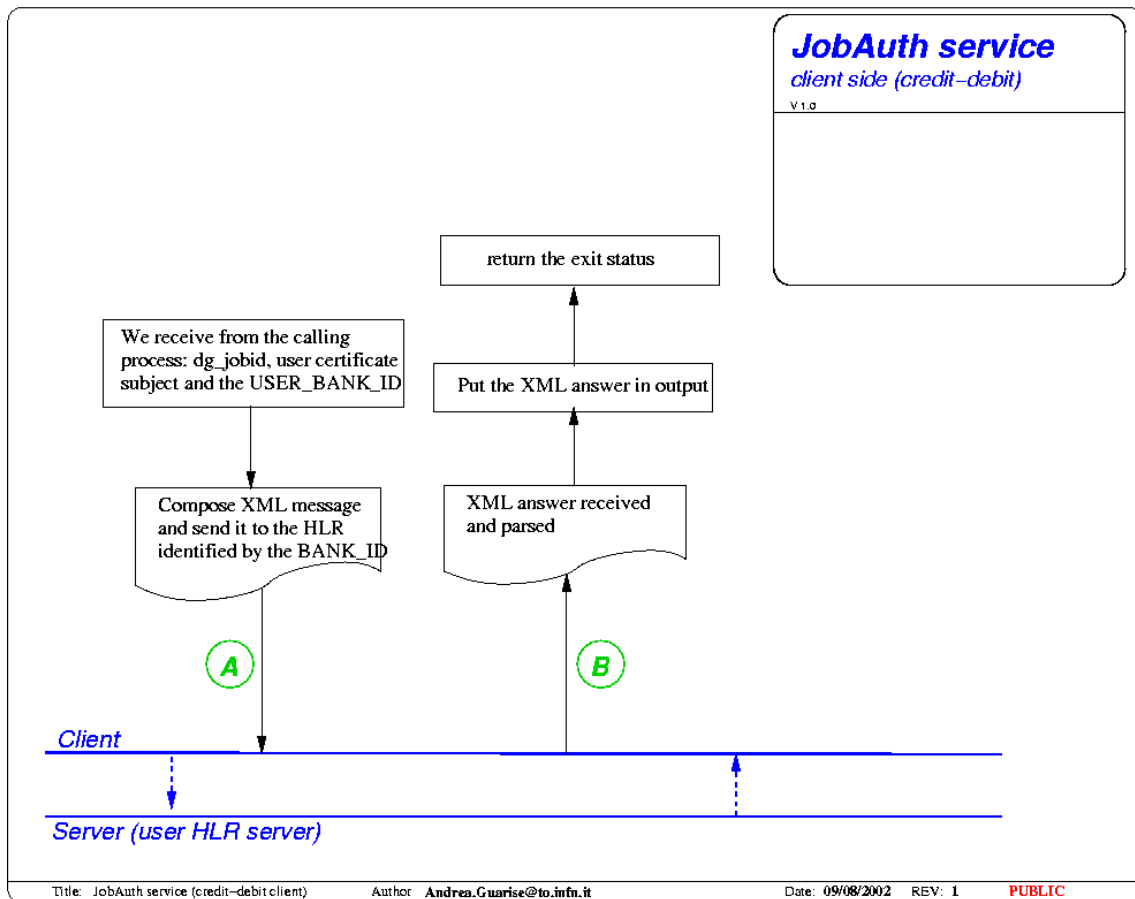


Figure 1: JobAuth client flowchart

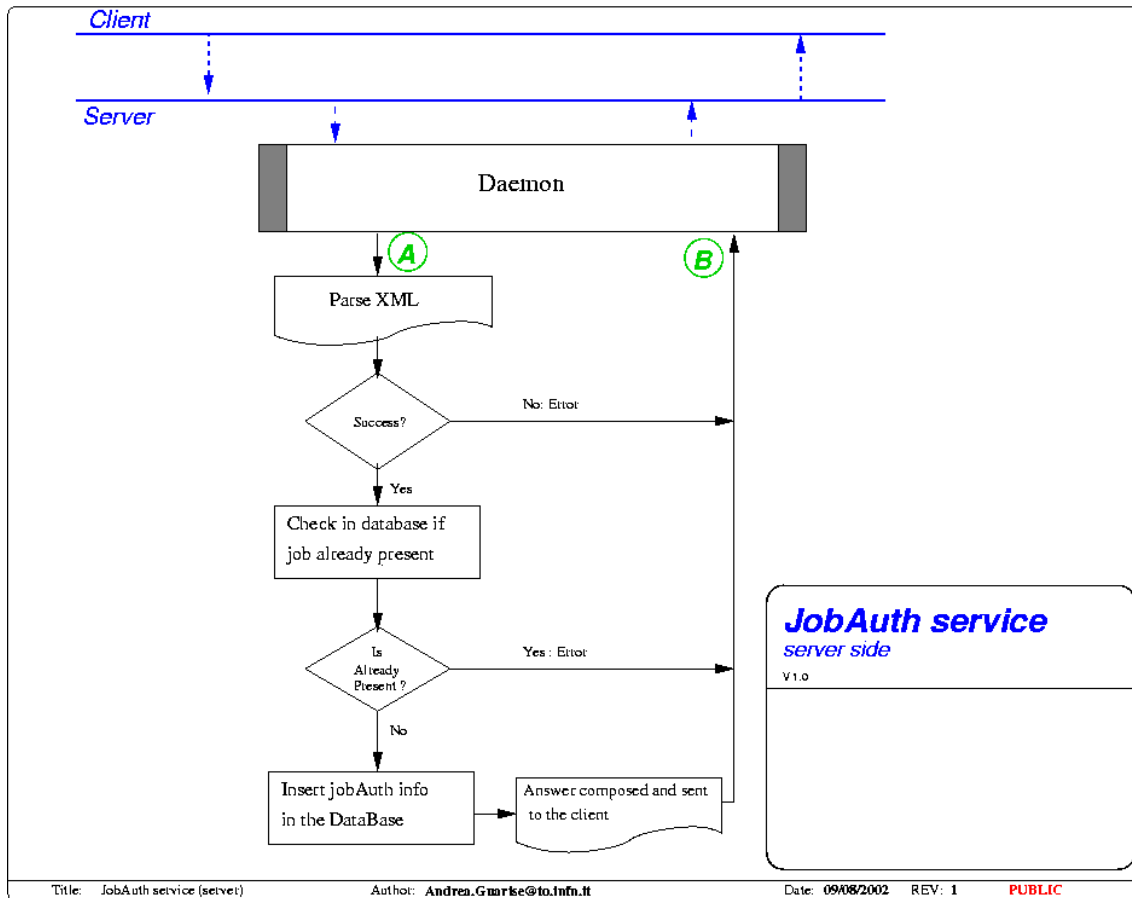


Figure 2: JobAuth server flowchart

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 debug 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>
```

3.1.2. UI - JobAuth service interaction diagrams

In Figure 3 we present the interaction diagram between the DataGrid User Interface and the DGAS JobAuth service. In the diagram is also present the interaction with the object 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.

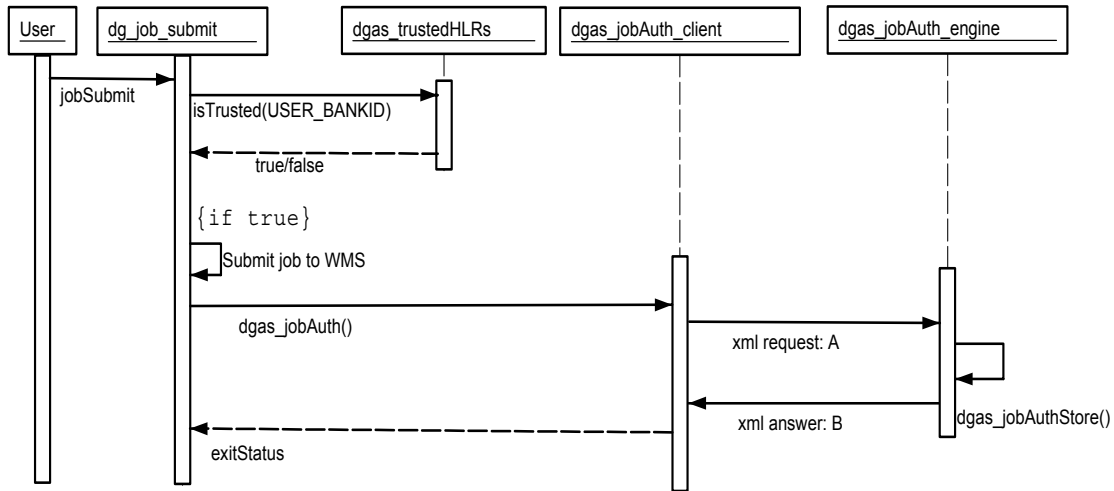


Figure 3: UI - JobAuth service interaction diagram

3.1.3. 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 4

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 toward the server is authenticated via GSI this information can be retrieved directly by the server, so it is optional.	Optional

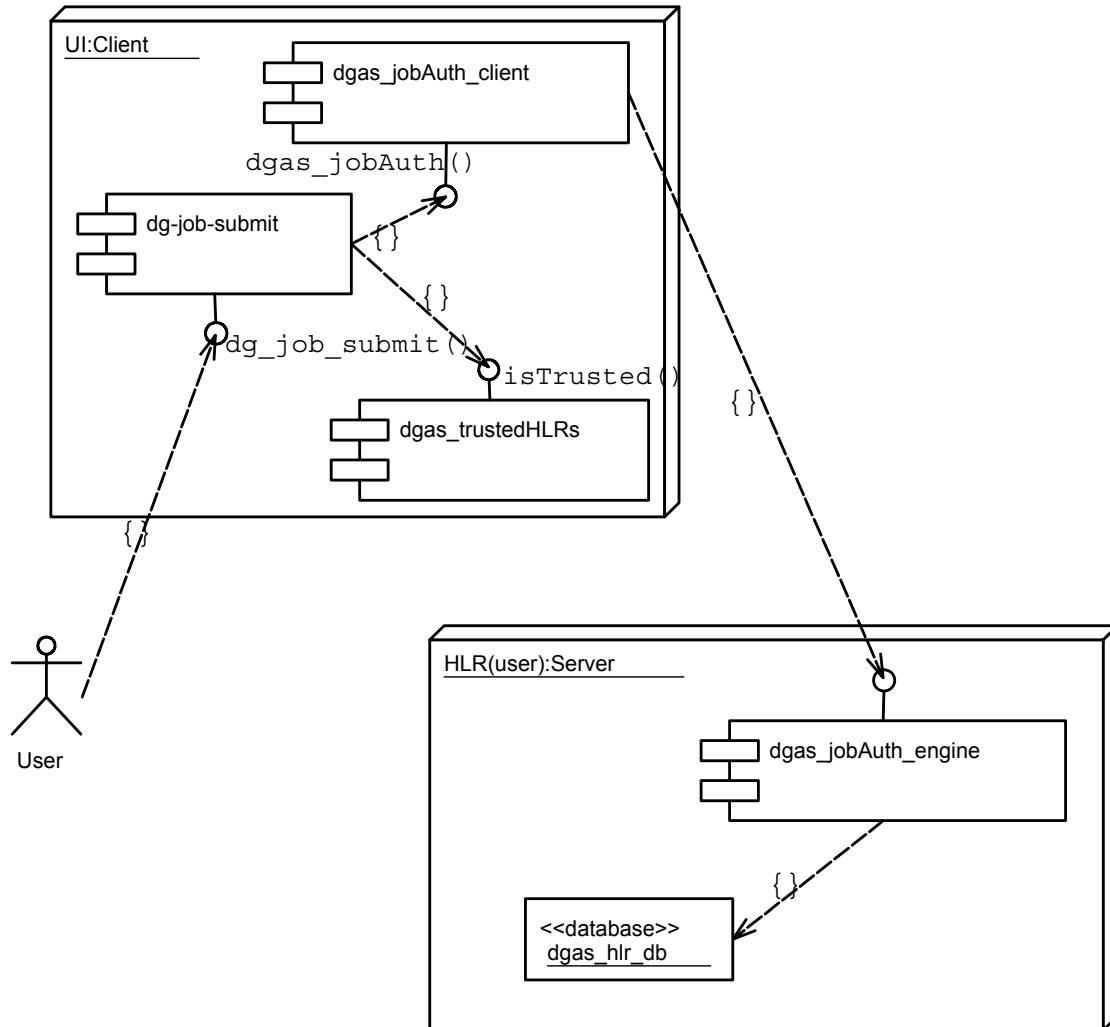


Figure 4: UI - JobAuth service deployment diagram

3.2. SERVICE API

3.2.1. C++

In this section two different services are covered, both with their APIs. The first is the job authorization service, whose calls are defined in:

```
$DGAS_INSTALL/include/jobAuth_client.hpp.
```

The information needed by the jobAuth client must be inserted in the structure:

```
struct jobAuth_data
{
    string dgJobId;
    string user_CertSubject;
```

```
    int submTime;  
};
```

and then communicated to the user HLR via the command:

```
int dgas_jobAuth(  
    string &user_acct_bank_id,    //ID for the user HLR, Mandatory  
    jobAuth_data &auth_data,    //Job authorization info  
    string *server_answer        //XML answer from the server  
);
```

Where the return code is 0 on success > 0 on failure and the string `server_answer` will contain the full XML answer from the server.

The `dgas_trustedHLRs` service should be very simple, essentially a function that will check the given `user_acct_bank_id` against a list of trusted banks contained in a plain text file. The file itself can be then updated automatically from a central repository.

The given function is:

```
Bool isTrusted(string user_acct_bank_id)
```

Where the string `user_acct_bank_id` is the user HLR that we want to verify.

3.2.2. Command line

We provide a command line program that can be used instead of the API. This can be useful in a script.

```
dgas_jobAuthclient <OPTIONS>
```

Where options are:

```
-h --help           :this help  
-j --jobid         :DgJobId of the current job  
-t --time          :Submission time of the current job  
-U --usrcert       :X509 subject of the user owning the job  
-u --usrhlrid      :Remote (user) ACCT_BANK_ID, HLR of the user.
```

Where `-t` and `-U` are optional. With the first you can specify the timestamp for the job submission time. The user cert option instead is used only for debugging since when the API's are build with the GSI API, the server side will automatically use the subject of the x509 certificate used to authenticate the connection. Thus in order to properly work in the accounting context, both the API and the command line tool must be used directly under the security context of the user.

4. RB – PA SERVICE

This service is the key of the economic brokering process. The idea is that the broker, after a preliminary selection of the resources matching a given job, selects the “right” resource according to a price assigned to the resource itself and a predefined economic policy, like for example : “chose the least expansive”. The price is assigned to the resources by a Price Authority.

There may be more PA spread over the Grid, so every resource belongs to a well defined PA. That PA will dynamically assign the price to that resource according to its status.

Here we describe the interface used by the RB to get the price from the resource’s PA.

More details on the PA itself can be found in R2.

4.1. SERVICE ARCHITECTURE

4.1.1. PA service flowcharts

In Figure 5 and Figure 6 we present the flowcharts of this service.

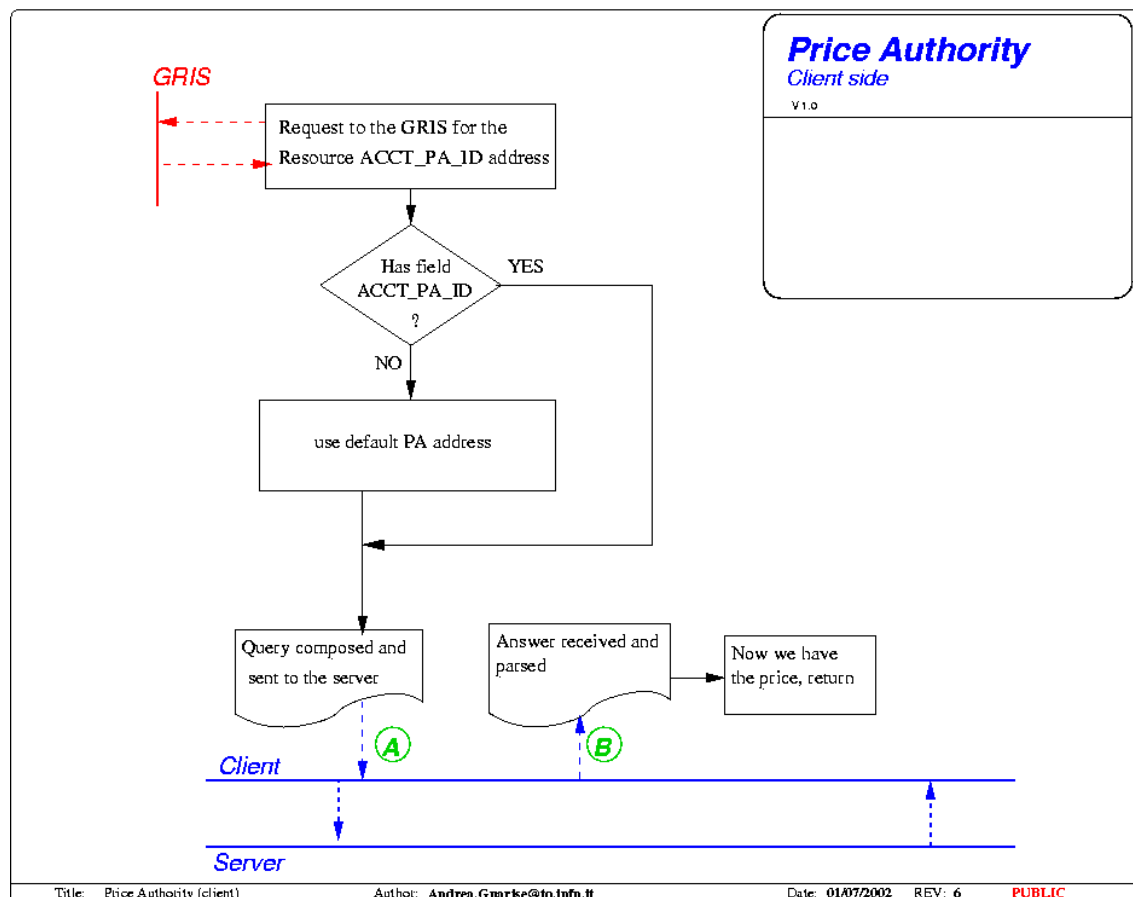


Figure 5: PA client flowchart

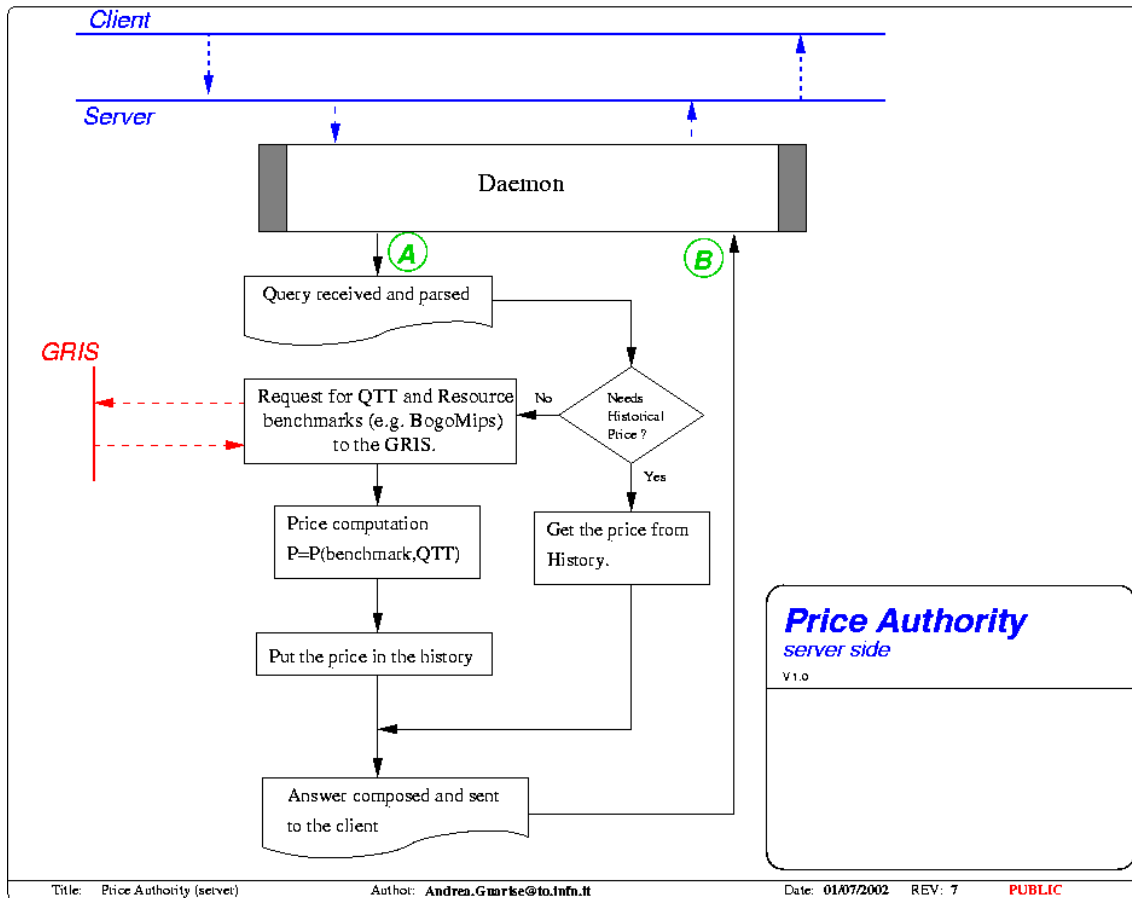


Figure 6: PA server flowchart

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>
      <RES_ID>
        resource certificate subject
      </RES_ID>
      <TIME>
        timestamp (in GMT) for the request
    </PRICE_INFO>
  </BODY>
</HLR>
  
```

```
</TIME>
</PRICE_INFO>
</BODY>
</HLR>
```

XML B, Answer from the server.

Description: The server retrieves the price 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>
    resource certificate subject
  </RES_ID>
  <MIN_TTL>
    validity time (seconds) for the resource price.
  </MIN_TTL>
  <PRICE>
    resource price (Grid Credits)
  </PRICE>
</PRICE_INFO>
</BODY>
</HLR>
```

When the price info can't be retrieved, the message contains an error, the body of the message will contain an entry like:

```
<PRICE_INFO>
  <RES_CERT_H>
    resource certificate subject
  </RES_CERT_H>
  <ERROR>
    ErrorCode
  </ERROR>
</PRICE_INFO>
```

4.1.2. RB - PA service interaction diagrams

In Figure 7 we present the interaction diagram between the DataGrid Resource Broker and the DGAS Price Authority service.

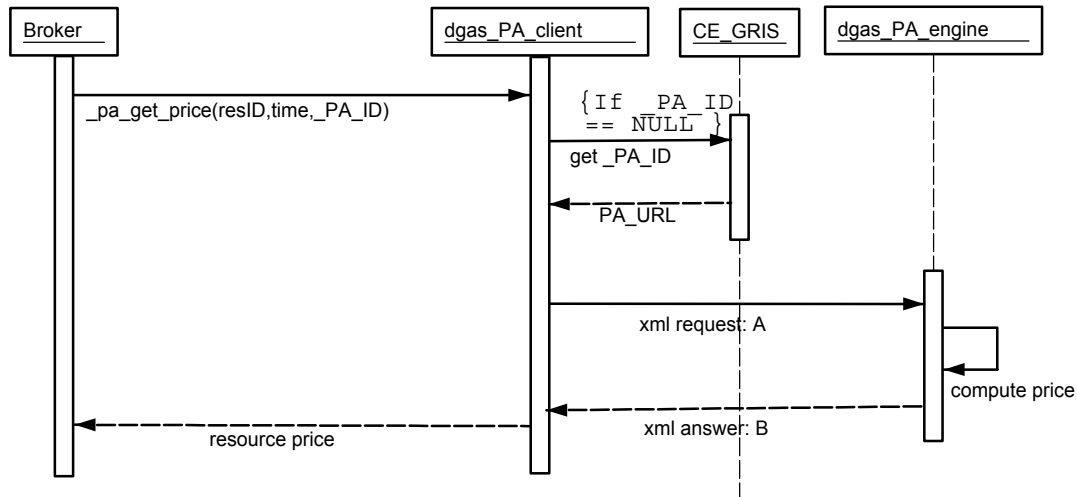


Figure 7: RB - PA service interaction diagram

4.1.3. 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 8

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

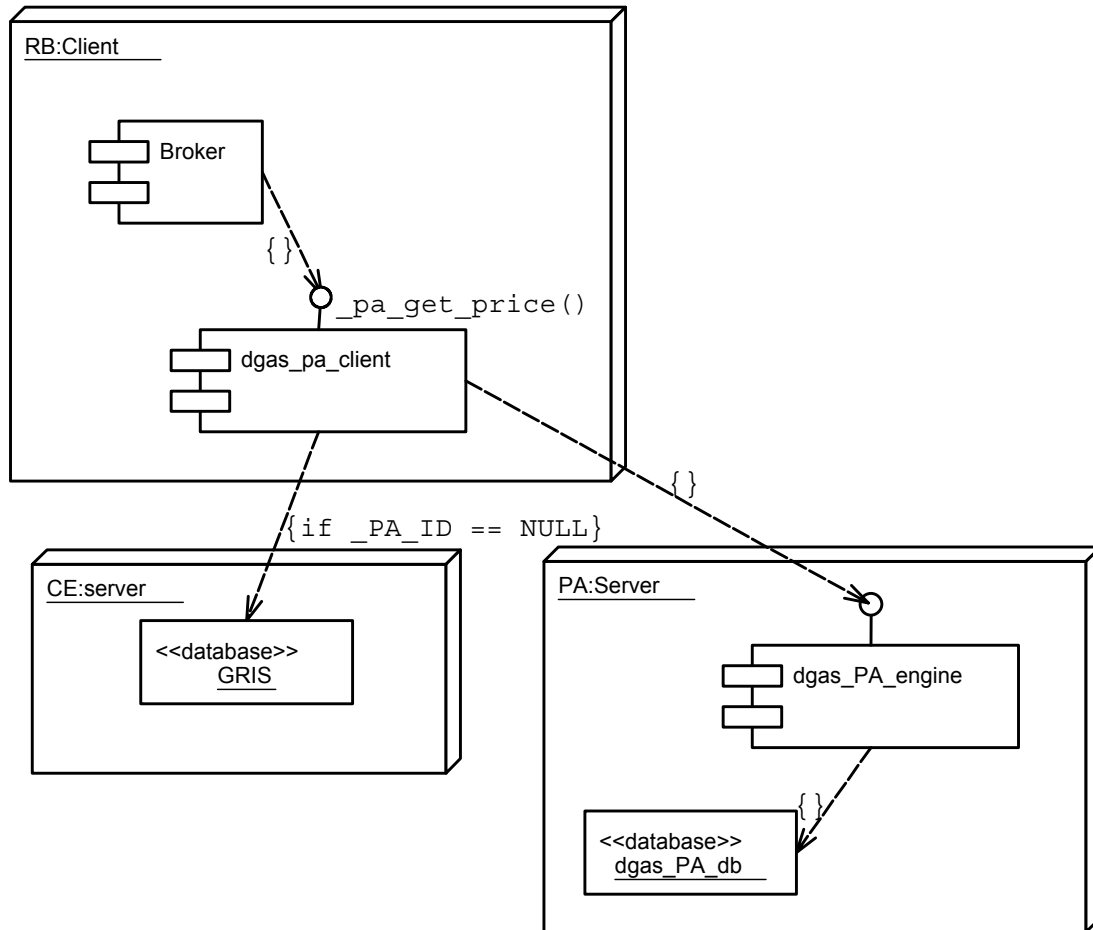


Figure 8: RB - PA service deployment diagram

4.2. SERVICE API

4.2.1. C++

In this section we cover the C++ bindings of the `dgs_pa_client` APIs defined in:

`$DGAS_INSTALL/include/dgas_pa_client.hpp`.

The information needed by the PA client must be inserted in the structure:

```
struct price
{
    string resID;    //identify the resource
    int time;      //GMT timestamp of the request;
    int price;     //resource price (GridCredits/Usage)
    int minTTL;   //minimum validity interval for the price (seconds)
};
```

and then communicated to the Price Authority via the command:

```
int  dgas_pa_client(  
    string &_acct_pa_id,    //ID for the Price Authority, Mandatory  
    price *res_price, //Job price info  
    );
```

Where the return code is 0 on success > 0 on failure.

On calling the function, the user needs to fill the `res_price` structure with the correct values for the `resID`, that is the identifier of the resource in the PA db, and `time`, which is the current timestamp.

The other fields are ignored and so can be set to NULL.

Once the function returns, if everything went fine, the `res_price` structure will contain the `price` and `minTTL` value of the needed price.

Even if the RB must always give the current timestamp in the `time` variable, if the client furnishes a timestamp in the past, the server will retrieve the price considered the valid one in that moment.

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 in this moment, but will be essential if a tighter policy will be adopted by the DataGrid community.

5.1. SERVICE ARCHITECTURE

5.1.1. HLR userAuth service flowcharts

In Figure 9 and Figure 10 we present the flowcharts of this service. The service is used by the Resource Broker to check the user job request against the economic authorization policy.

Note that currently an user request will be always authorized in order to help the users gain confidence with the system.

The cost algorithm used in the client to estimate the job cost is the same used in 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.

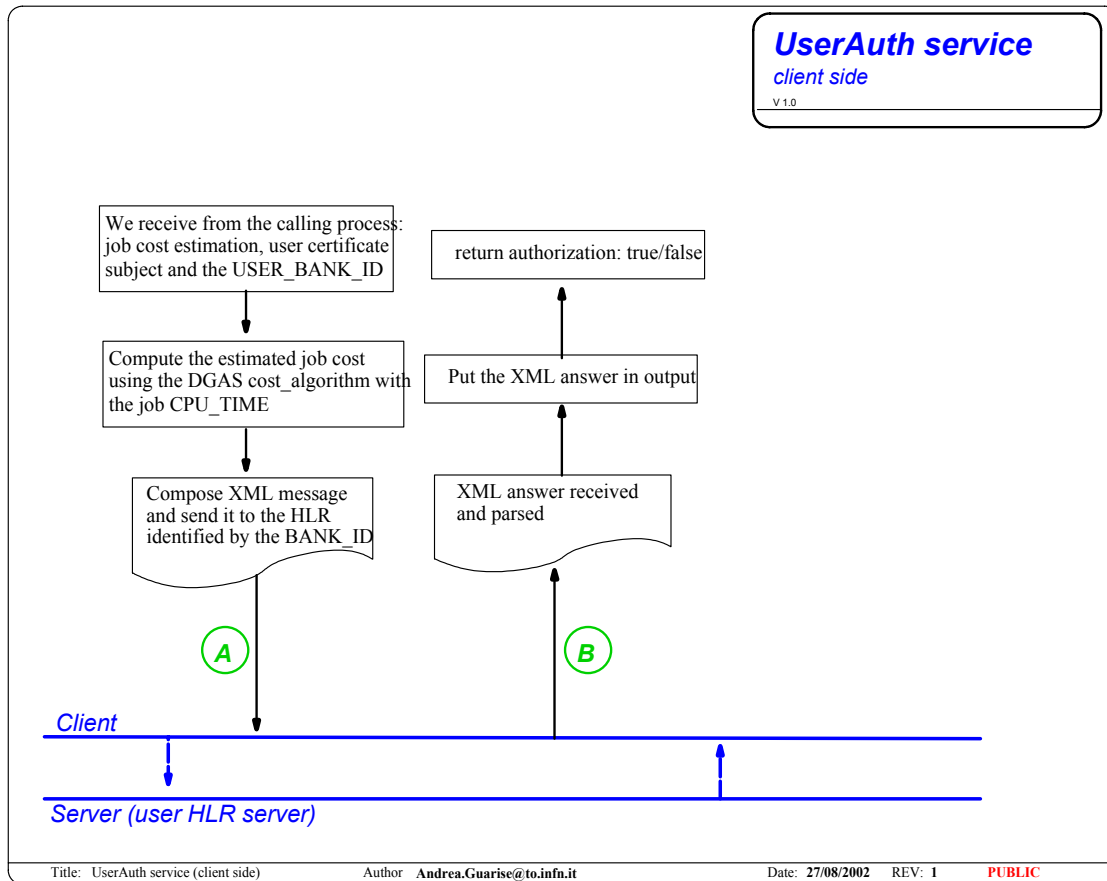


Figure 9: UserAuth service, client side

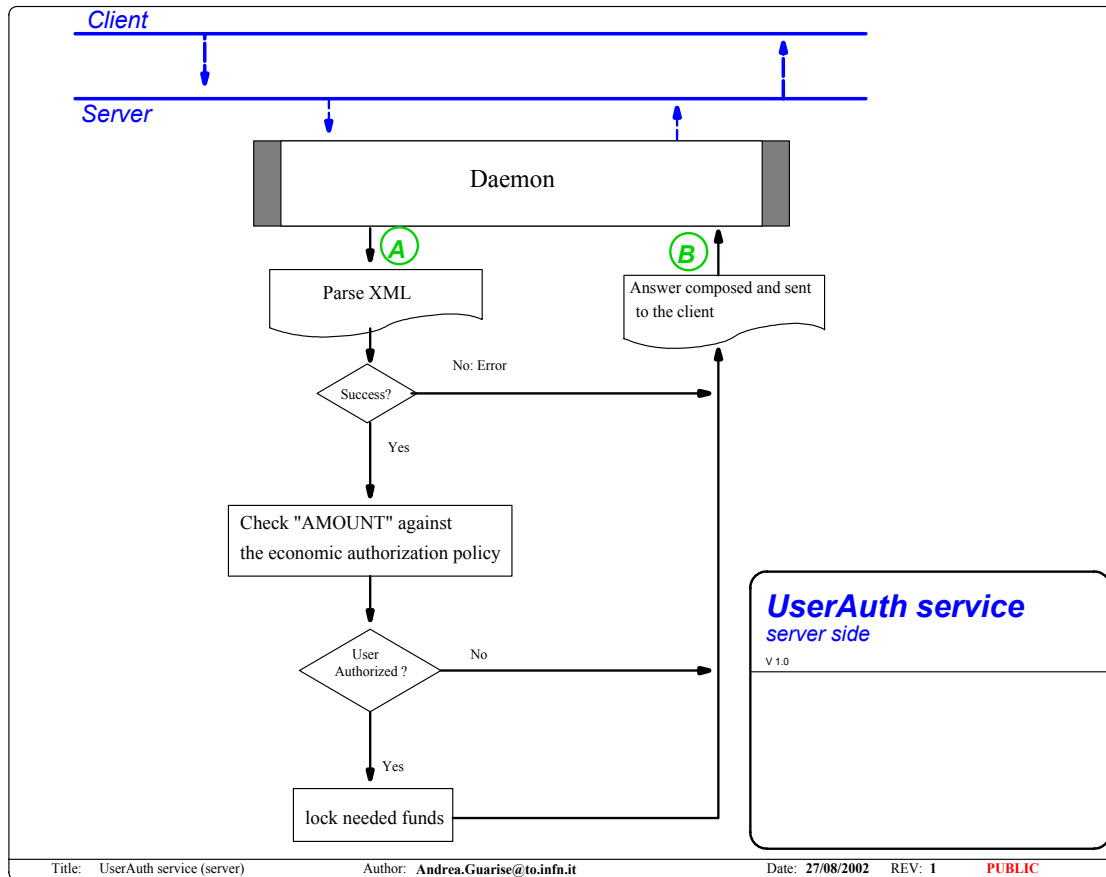


Figure 10: UserAuth service, server side

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="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>
  
```

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

XML B, Answer from the server.

Description: The server retrieves the price of the resource, or computes it if necessary, and sends it to the client.

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>
```

5.1.2. RB - HLR service interaction diagrams

In Figure 11 we present the interaction diagram between the DataGrid Resource Broker and the DGAS HLR user economic authorization service.

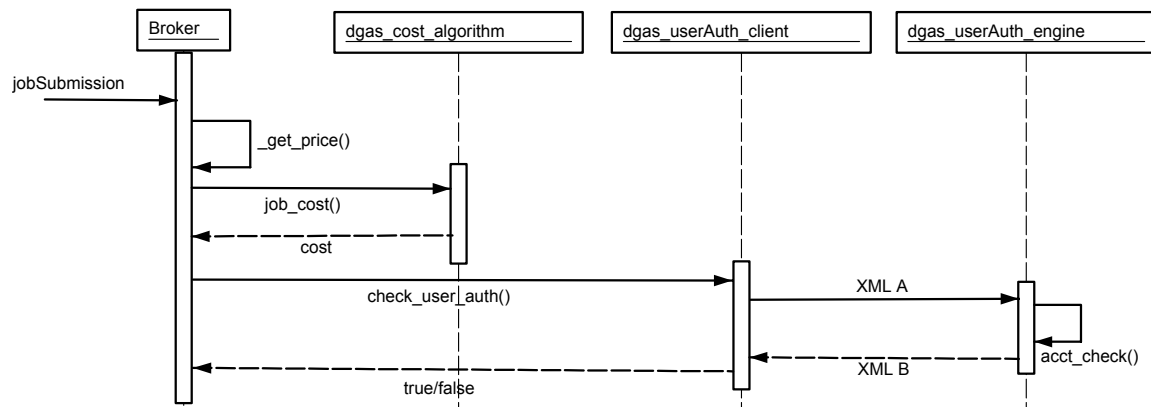


Figure 11: RB-HLR user authorization interaction diagram

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).

5.1.3. 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 12.

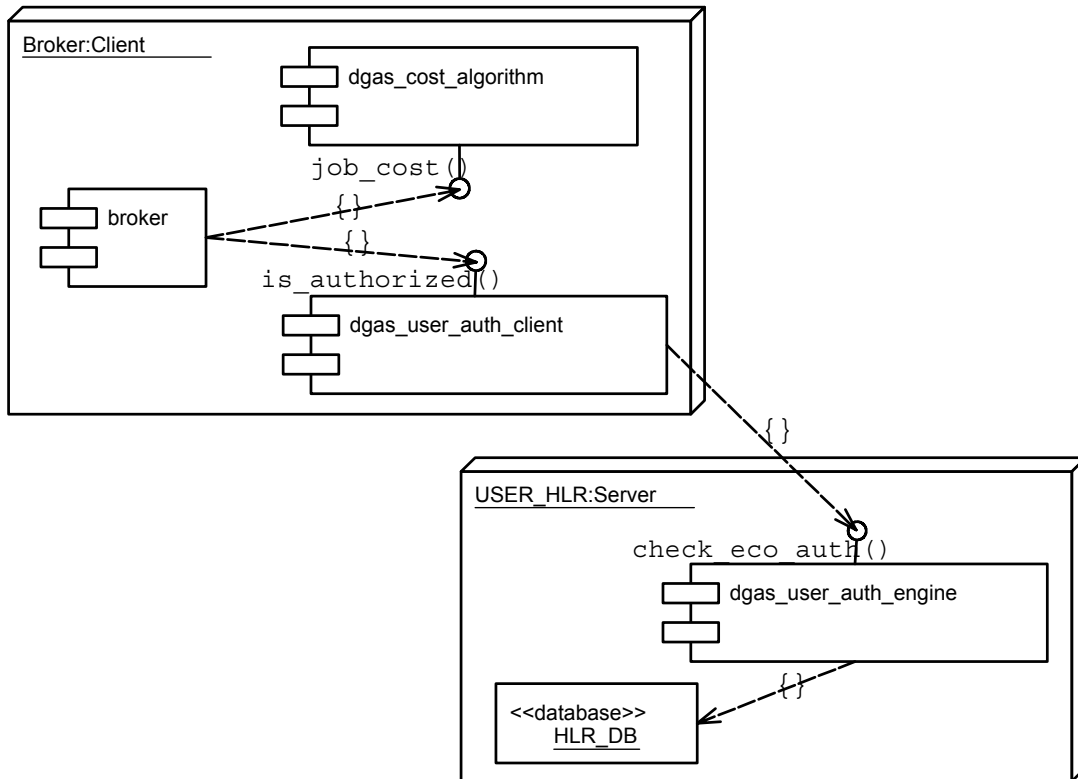


Figure 12: RB-HLR user authorization deployment diagram

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 one Hour.	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



5.2. SERVICE API

5.2.1. C++

`$DGAS_INSTALL/include/dgas_userAuth_client.hpp.`

6. 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 here we want to explain.

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 identified by their X509 certificate subject.

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 job¹ it sends the following information to the user HLR (See Figure 13), 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**
Resource X509 certificate Subject	Mandatory

¹ 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.

*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 happen 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.

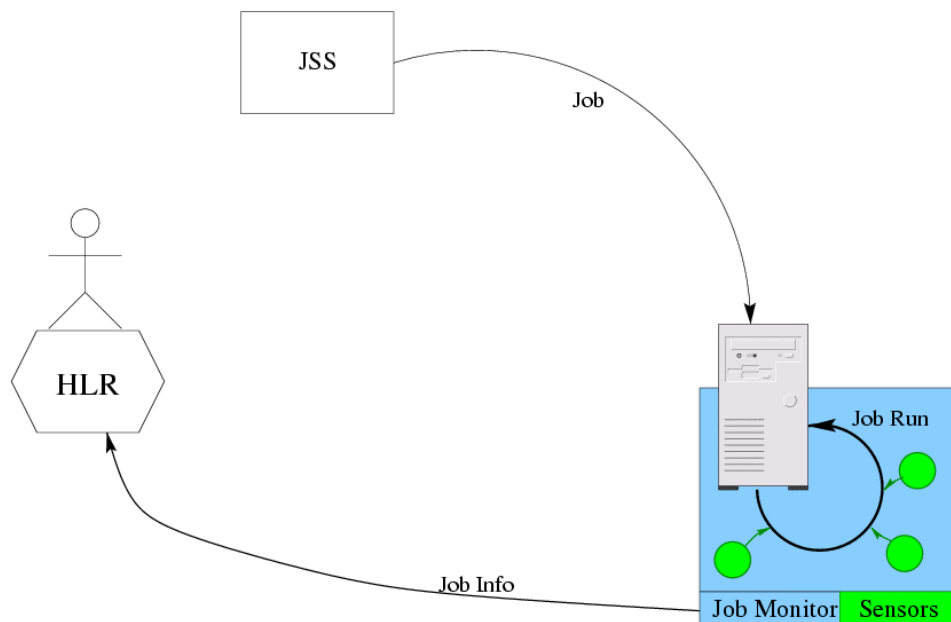


Figure 13 JobMonitor and sensors scheme

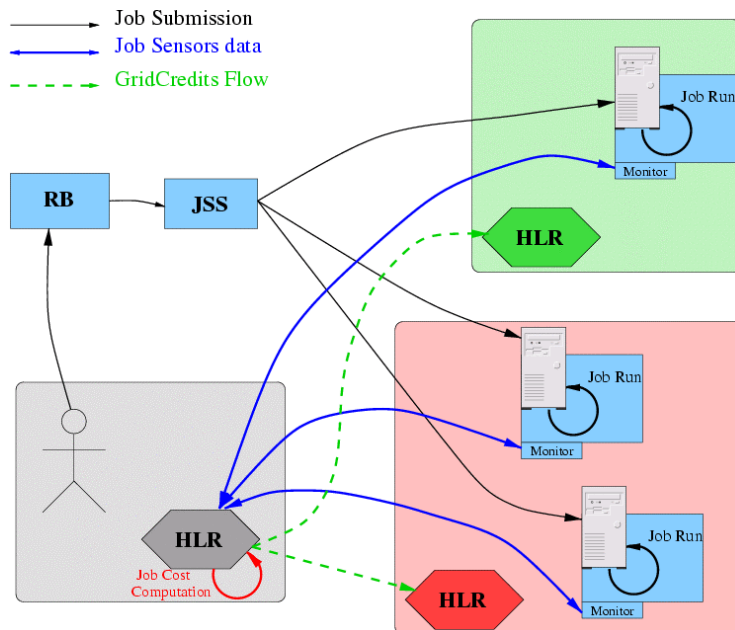


Figure 14 Job Monitor and cost computation data flow

Figure 14 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.

6.1. SERVICE ARCHITECTURE

6.1.1. ATM Service flowcharts

Figure 15 and Figure 16 represent the Computing Element Sensors client and server, respectively. This service is activated when a job terminates. The client combines the JSS information with the consumption measurements furnished by the executing CE as input to the server that calculates the total cost. The price used is that 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.

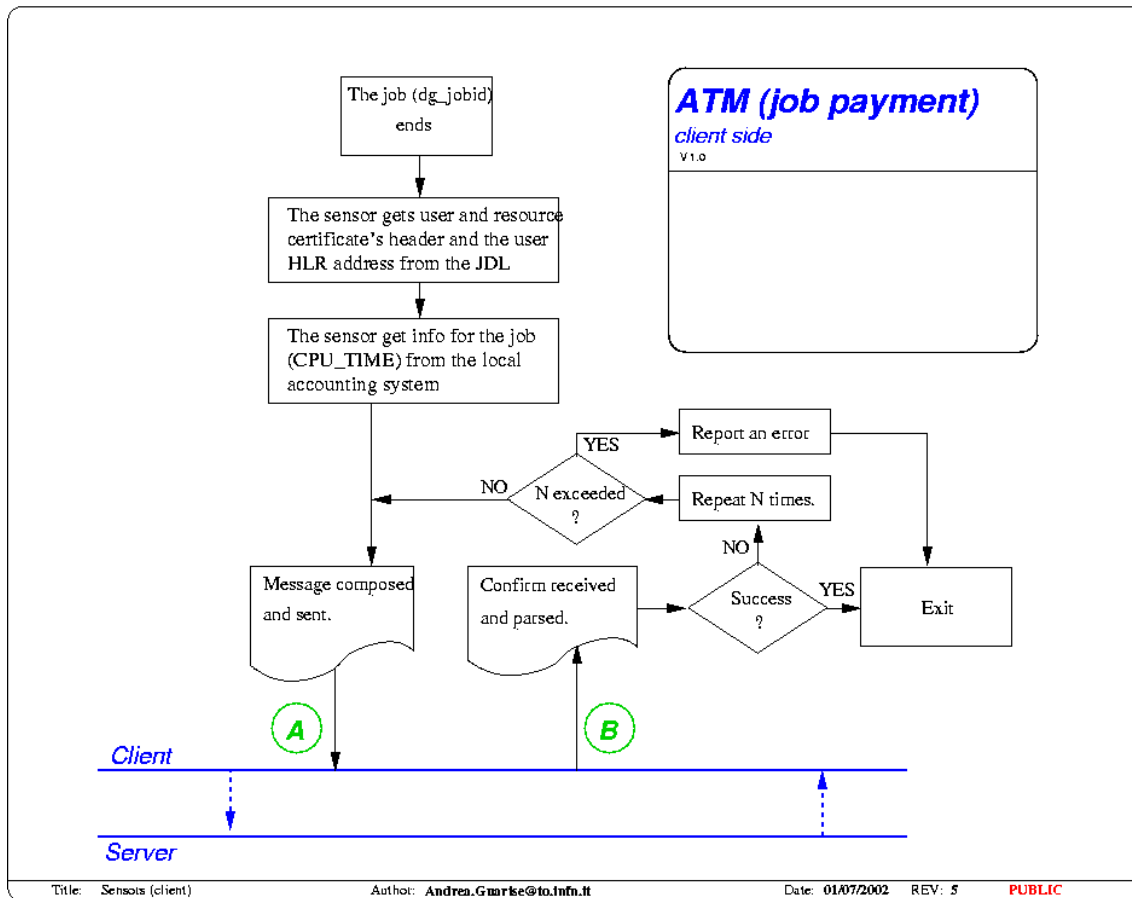


Figure 15: ATM (Job payment), client side.

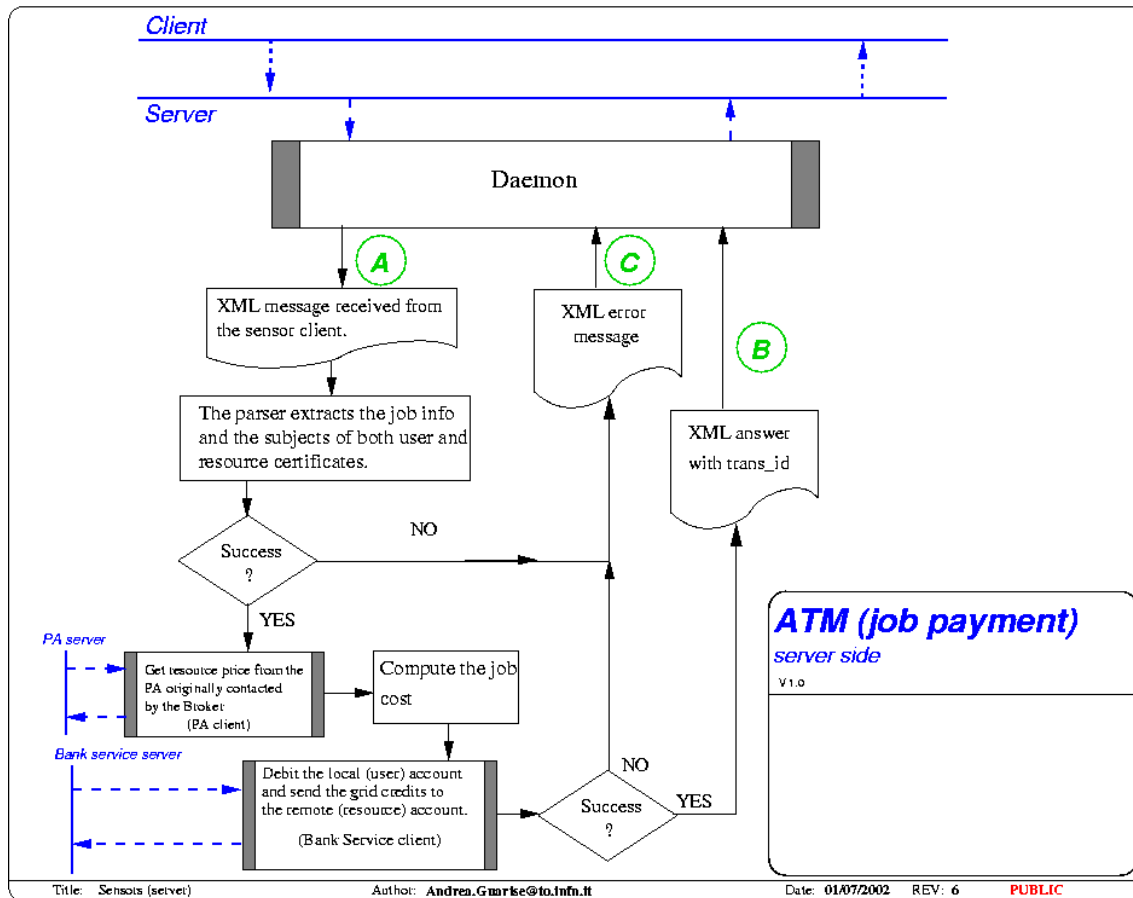


Figure 16: ATM (Job payment), server side.

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_certsubject 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_certsubject 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>
```

6.1.2. Sensor –ATM_service Interaction diagram

In Figure 17 we describe the current interaction diagram between the CE sensor and the ATM-engine on the DGAS User HLR server. This architecture has clearly some disadvantages, mainly due to the fact that the ATM_engine processes the transaction synchronously as soon as the job information is available. This process, although simple can result in the sensor system waiting for the crediting process to finish correctly.

A better approach to this problem is illustrated in Figure 18, where the job information is stored in a queue and then processed asynchronously later. In this way the sensor system needs only to await the confirmation that the data were correctly received by the ATM service. This is clearly a better approach but requires more effort in the development phase.

However the public interfaces are the same in both cases, so the implementation of the second version remains a future issue.

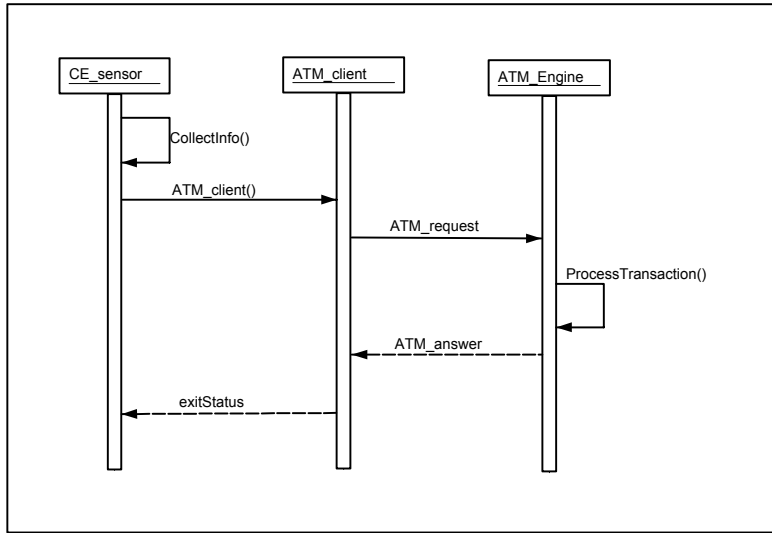


Figure 17: Sensor - ATM engine interaction diagram V1

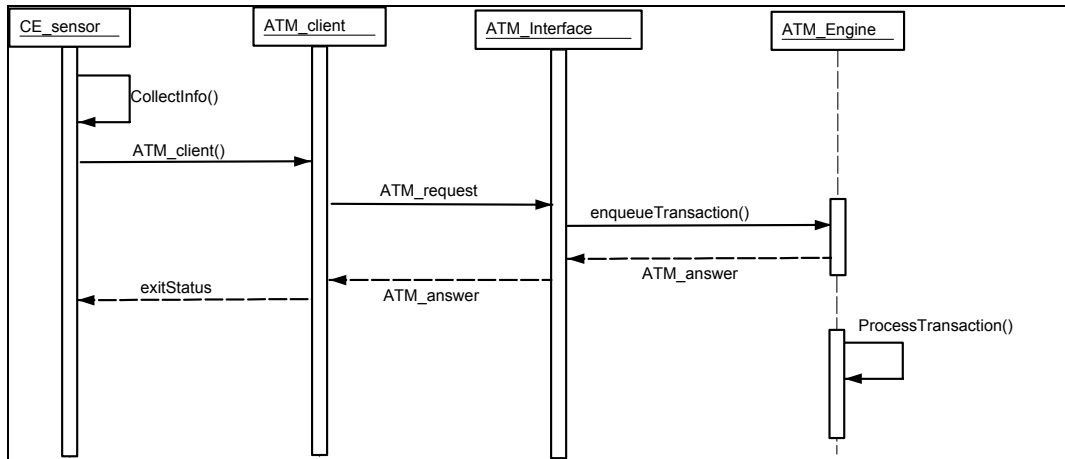


Figure 18: Sensor - ATM engine interaction diagram V2

6.1.3. Sensor –ATM_service deployment diagram

Figure 19 and Figure 20 represent the deployment structure of the relevant sensor and accounting components. Both the current and the future architecture are illustrated.

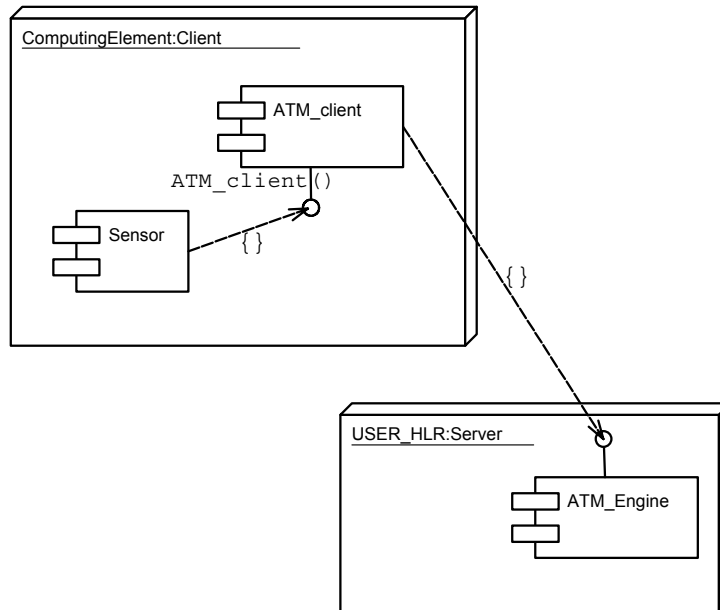


Figure 19 Deployment diagram V1

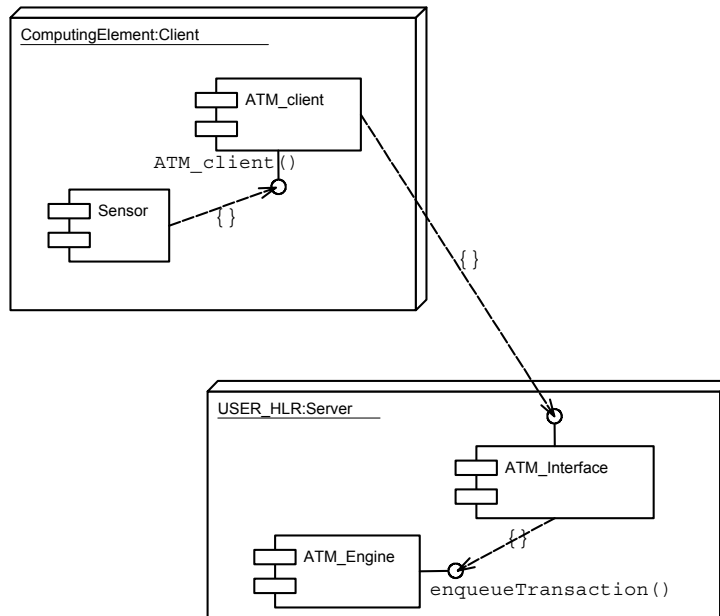


Figure 20: Deployment diagram v2

6.2. SERVICE API

We currently provide two means of using the ATM_client. The first is a C++ function (ATM_Client() in the diagrams), the second is a command line tool, intended for use within a script.

6.2.1. C++

The information retrieved by the sensor must be used to populate the following two structures defined in \$DGAS_INSTALL/include/ATM_client.hpp:

```
struct ATM_job_data
{
    string dgJobId;
    int time;
    string res_acct_PA_id;
    string res_acct_bank_id;
    string user_CertSubject;
    string res_CertSubject;
};
```

```
struct ATM_usage_info
{
    int cpu_time;
    int wall_time;
};
```

and then communicated to the user HLR via the command:

```
int ATM_client(
    string user_acct_bank_id, //ID for the user HLR, Mandatory
    ATM_job_data &job_data, //Job info
    ATM_usage_info &usage_info, //Usage records
    string *server_answer //XML answer from the server
);
```

Where the returncode is 0 on success > 0 on failure.

6.2.1.1. Command line

The command line tool is simply an interface to the previous function and can be also seen as an example on how to use the previous C++ API.

The command is atm_clientTest and can be found in \$DGAS_INSTALL/libexec/

```
ATM_client <OPTIONS> USAGE_RECORDS
```

Where options are:

```
-h --help           :this help
-j --jobid          :DgJobId of the current job
-t --time           :Submission time of the current job
-p --paid           :Resource ACCT_PA_ID, price authority for the CE
```



-l --localbankid :Local (resource) ACCT_BANK_ID, HLR for this CE
-C --cecert :X509 subject of this CE
-U --usrcert :X509 subject of the user owning the job
-r --remotehlrid :Remote (user) ACCT_BANK_ID,HLR of the user.

7. ANNEXES

7.1. 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
xerces	1.7.0	XML parser	

The software has been developed under Linux RedHat 7.2, other versions may work properly, but we didn't perform any test.