Contribution ID: **61** Type: **Oral contribution**

Development of gLite Web Service Based Security Components for the ATLAS Metadata Interface

Thursday 2 March 2006 14:20 (20 minutes)

Introduction

AMI (ATLAS Metadata Interface) is a developing application, which stores and allows access to dataset metadata for the ATLAS experiment. It is a response to the large number of database-backed applications needed by an LHC experiment called ATLAS, all with similar interface requirements. It fulfills the need of many applications by offering a generic web service and servlet interface, through the use of self-describing databases. Schema evolution can be easily managed, as the AMI application does not make any assumptions about the underlying database structure. Within AMI data is organized in "projects". Each project can contain several namespaces (*). The schema discovery mechanism means that independently developed schemas can be managed with the same software.

This paper summarises the impact of the requirements contracted to AMI of five gLite metadata interfaces. These interfaces namely MetadataBase, MetadaCatalog, ServiceBase, FASBase and MetadaSchema [1] deal with a range of previously identified use cases on dataset (and logical files) metadata by particle physicists and project administrators working on the ATLAS experiment. The future impact on AMI architecture of the VOMs security structure and the gLite search interface are both discussed.

Fundamental Architecture of AMI

The AMI core software can be used in a client server model. There are three possibilities for a client (software installed on client side, from a browser and web services) but the relevant client with regards to grid services is the Web Services client.

Within AMI there are generic packages, which constitute the middle layer of its three-tier architecture. Command classes can be found within these packages. These classes are key to the implementation of the gLite methods in each of the interfaces. The implemented gLite interfaces are therefore situated on the server side in this middle layer and directly interface with the client tier and the command classes in this middle layer. It is possible to choose a corresponding AMI command that is equivalent to the basic requirements of each of the gLite Interface methods.

[Figure 1]

Figure 1: A Schematic View of the Software Architecture of AMI [2]. This diagram shows the AMI Compliant Databases as the top layer. This interfaces with the lowest software layer, which is JDBC. The middle layer BkkJDBC package allows for connection to both MySQL and Oracle. The generic packages contain command classes which are used in managing the databases. Application specific software in the outer layer can include the generic web search pages.

The procedure used to further understand the structure necessary to implement the gLite methods was to observe how AMI is designed to absorb commands into its middle tier mechanism. This was achieved by mapping the delegation of methods through the

relevant code and is best illustrated with the use of an UML sequence diagram in figure 2.

The deployment of AMI as a web application in a web container can take place using Tomcat. To set up web services for AMI it is necessary to plug the Axis framework into Tomcat. Then with the use of WSDL and the axis tools that allow conversion from WSDL to Java client classes a Java web service client class can be deployed which communicates with the gLite interfaces.

(*) namespace is “database” in MySQL terms, “schema” in ORACLE and “file” in SQLite.

[Figure 2]

Figure 2: UML sequence diagram of basic workings of AMI. Note: A controller class delegates what command class is invoked. A router loader is instantiated to connect to a database. XML output is returned to the gLite interface implementation class.

A direct consequence of grid services is secure access. This involves authentication and authorisation of users and machines. Authorisation in AMI is handled by a local role-based mechanism. Authentication is implemented by securing the web services using grid certificates.

Currently permissions in AMI are based on a local role system. An EGEE wide role system called Virtual Organizations Membership Service (VOMS) [3] is being developed. AMI would then have to be set up to read and understand VOMS attributes and grant permissions based on a user’s role in ATLAS. Requirements analysis work is currently underway on the impact of this VOMS system on the AMI architecture.

Also directly relevant to the gLite interface was the implementation of a query language for performing cascaded searches through all projects. This implementation used a library (JFLEX) to define our own grammar rules, following the EGEE gLite Metadata Query Language (MQL) specification. It allows AMI to execute a search in a generic way on several databases of any type (MySQL, ORACLE or SQLite for example) starting only from one MQL query.

Conclusion

This paper presents a description of the implemention of the gLite Interfaces for AMI. It summarises how AMI was set up fully with these implementation classes interfacing with web service clients and how these clients are made secure with the aid of grid certificates.

AMI as mentioned provides a set of generic tools for managing database applications. AMI also supports geographical distribution with the use of web services. To implement the gLite interfaces as a wrapper to AMI using these web services provides the user with a generic and secure metadata interface. Along with the gLite search interface, any third party application should be able to plug in AMI knowing it supports a well defined API.

References

[1] Developer’s Guide for the gLite EGEE Middleware - http://edms.cern.ch/document/468700
[2] ATLAS Metadata Interfaces (AMI) and ATLAS Metadata Catalogs, Solveig Albrand, Jerome Fulachier, LPSC Grenoble
[3] VOMs - http://hep-project-grid-scg.web.cern.ch/hep-project-grid-scg/voms.html

Primary author: Mr DOHERTY, Thomas (University of Glasgow)

Co-authors: Mr LAMBERT, Fabian (IN2P3); Mr FULACHIER, Jerome (IN2P3); Ms ALBRAND, Solveig (IN2P3)

Presenter: Mr DOHERTY, Thomas (University of Glasgow)

Session Classification: 2b: Data access on the grid

Track Classification: Data access on the grid