The ATLAS metadata interface

Solveig Albrand, Jérôme Fulachier and Fabian Lambert

Laboratoire de Physique Subatomique et Corpusculaire, Université Joseph Fourier Grenoble 1, CNRS/IN2P3, INPG, 53 avenue des Martyrs, 38026, Grenoble, FRANCE

Email: solveig.albrand@lpsc.in2p3.fr

Abstract. AMI is the main interface for searching for ATLAS datasets using physics metadata criteria. AMI has been implemented as a generic database management framework that allows parallel searching over many catalogues, which may have differing schema, and may be distributed geographically, using different RDBMS.

The main features of the web interface will be described; in particular the powerful graphic query builder. The use of XML/XLST technology ensures that all commands can be used either on the web or from a command line interface via a web service.

1. Introduction

This article describes the ATLAS metadata interface (AMI) [1] which was chosen to be the official ATLAS tool for dataset selection in July 2006. We have previously described the context of metadata in ATLAS and the place of the dataset search within this context [2]. An overview of ATLAS metadata has been presented at this conference [3]. In this article we give more detailed information concerning the architecture of AMI. The details of some of the advanced features of the dataset search web interface are described. We discuss some recent work to improve the quality of service.

2. Main features of the AMI framework

AMI is a framework consisting of relational databases which contain their own description, a software layer to exploit this auto-description, and a set of generic interfaces for the usual database functions (insert, update, delete and select). AMI can manage different database schema deployed on geographically distributed servers with different relational database management systems (RDBMS) simultaneously. This enables AMI to support schema evolution in a seamless way.

The AMI framework supports several applications for ATLAS:

- The Dataset Search
- The Tag Collector
- Nomenclature reference tables
- The Monte Carlo dataset number broker
- Catalogues of nightly builds.

The AMI server code is written in JAVA and uses JAVA Database Connections (JDBC). In consequence, it is independent of platform, operating system and database technology. The application software is built in three levels above JDBC as shown in Figure 1. The three lower level packages wrap the connections to the databases managing transactions, connection pooling, transmission of SQL commands, and recuperation of query results. The top layer of the software contains application specific packages with knowledge of non generic features of the database tables. Those which are for dataset searching contain in particular the software to manage all the tasks that pull data from the various sources, and the dataset provenance tracking.

OVERVIEW of AMI Software			
Application Specific Software	DATASET SEARCH	TAG COLLECTOR	OTHERS
AMI Generic Software	Generic AMI Command management		
Database Connection management	Connection Pooling, Transaction management of AMI Commands		
JDBC	Standard JAVA Package		
Specific Database Libraries	ORACLE	mySQL	*SQLITE*

Figure 1 : The layered architecture of AMI software

A client who queries an AMI database is not expected to have knowledge of the name or type of the database, or the name of any database tables, or the relation between the database tables. The architecture allows clients to express queries in terms of the application semantics. Thus a user of the AMI dataset selection application should be able to work with a schema of datasets and files, whereas a user of another AMI based application, such as Tag Collector, works with different semantics.

3. Authorisation, Authentication and User Specific Functions

AMI implements several mechanisms of user authentication:

- Username and Password.
- Grid Certificate.
- Possibility to upload a VOMS [4] proxy (using ACACIA [5]) and to pass this proxy to a third party application. This mechanism is shown in Figure 2. It is used notably for the formation of "super datasets" for instance the reprocessing coordinator may make a container dataset which groups together all the files in other containers which are known to be good. Since the ATLAS Distributed Data Management (DDM) only allows one level of hierarchy this must be done with a special interface. Such an interface has been implemented in AMI, and uses the VOMS proxy of the reprocessing coordinator to make the new containers. AMI does not need any special dataset making rights, merely transmitting VOMS proxy of the AMI user to the DDM client.

It is important to note that users of AMI do not possess a real database password. All DB access is controlled at the application level.

Authorization is managed by a hierarchical set of roles.

- Users are mapped to one or more roles within the AMI framework.
    - John Smith is a Monte Carlo Coordinator.
    - Jean Dupont is a Package manager of the Package XYZ.
- Roles are mapped to commands.
    - A Monte Carlo Coordinator can edit the cross section information of any Monte Carlo Datasets.
    - Package Managers can add new code modules to their package.
    - A "guest" can only read.
- Further restriction is possible by assigning a "validator" class to the role.
    - Jean Dupont can only add new code modules to Package XYZ.
    - Jane Doe can only update information on records which she herself added.

Roles are organized in hierarchies, so that a Manager can give rights to a part of his domain to someone else.

There are also AMI roles in ATLAS VOMS, for example "AMI Writer". This possibility is not exploited as yet, but it would give the possibility for ATLAS management to regulate who can write to AMI.

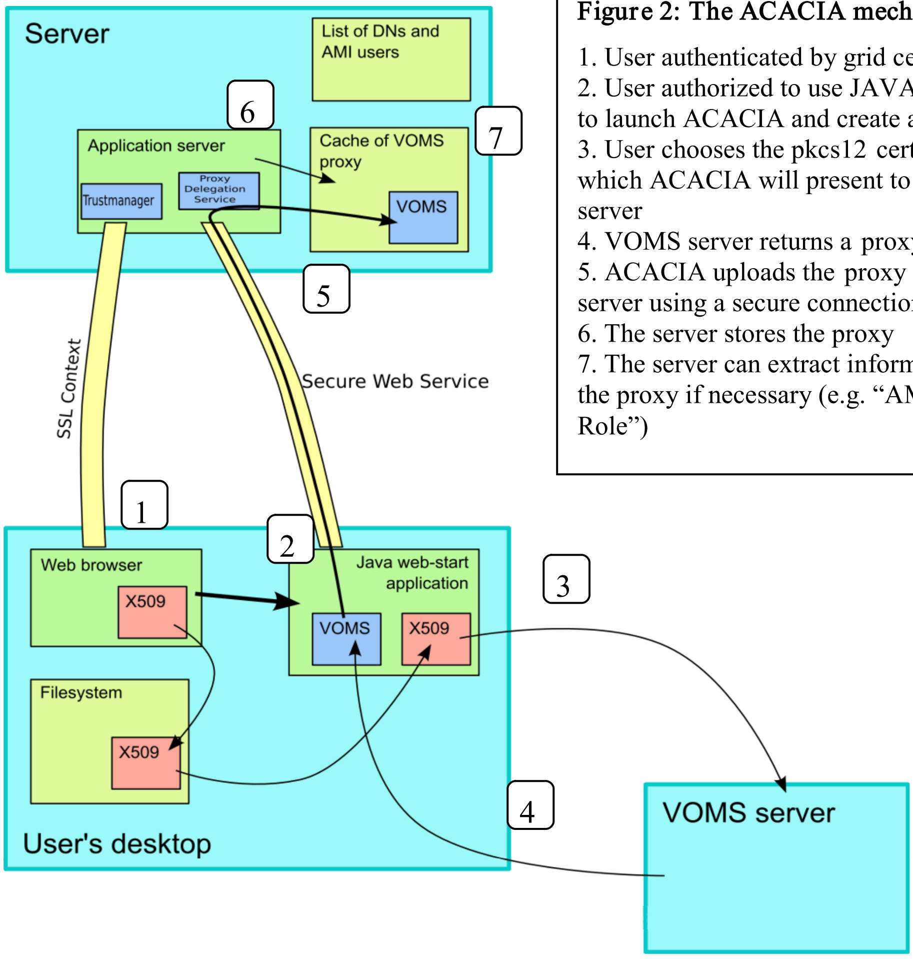


Figure 2: The ACACIA mechanism

1. User authenticated by grid certificate
2. User authorized to use JAVA Web Start to launch ACACIA and create a proxy
3. User chooses the pkcs12 certificate which ACACIA will present to the VOMS server
4. VOMS server returns a proxy
5. ACACIA uploads the proxy on the server using a secure connection
6. The server stores the proxy
7. The server can extract information from the proxy if necessary (e.g. "AMI Writer Role")

When a user logs on to AMI, all the web pages are modified to reflect his or her authorised actions. For example, a user who has the rights to update dataset information will see a button for this action. Other users who do not share this right will not see the button. Each user has access to a personalized home page. The home page is to list the user's bookmarks, and the roles assigned to the user. It is possible to limit the actions or even the visibility of certain fields to a particular role.

4. The AMI web interface

The AMI web interface portal page [1] gives users access to the dataset search, to their personal page, to related AMI applications such as the Monte Carlo dataset number broker, and it is also possible simply to browse all the data in AMI. In the rest of this section we discuss only the Dataset Search. Several types of search have been implemented:

- A simple search which performs a partial string search on the dataset name.
- A simple search which performs a keyword string search on a selection of fields.

- A more complex search which allows the user to specify the required values for a number of key parameters. This interface was defined with the help of one of the ATLAS user groups.
- A "hierarchical" search for datasets, where the user selects one parameter at a time.

Behind the scenes, user requests are transformed to a request expressed in the EGEE gLite grammar [6]. This grammar, which is also implemented by the native EGEE metadata application [7], looks similar to SQL but makes no assumptions about schema, so is ideally suited to the AMI philosophy. The gLite query is then sent out in parallel to all open dataset catalogues. They respond if they can reply to the query. (A query for information about a particular parameter can only be satisfied by a catalogue which contains information on this parameter.) A second request to the catalogue gets the data, which is then displayed for the user. If no datasets are found in the open catalogues the archived catalogues are searched, and if a result is found the user is warned that it comes from the archives. The archives can be included in the first search if the user desires.

The results are displayed for all catalogues. The user can modify the appearance of the results by selecting the fields to display, or the order in which they are displayed. Pop-up windows give additional information about the meaning of the fields. The results can be refined by added schema specific filters. Users can bookmark their favourite requests, which are always re-executed dynamically. It is also possible to have an RSS feed for a dataset record, to be informed when there is a modification in the state, for example when events become available.

4.1. The database schema for datasets

As explained above, AMI uses self-describing relational schema. The generic AMI commands make no assumptions about the schema; they use the internal description to discover it. On the other hand, the top layer software contains detail of the particular application's semantics. In the case of the dataset catalogues a few semantic constraints are imposed. All schema must have a database table called "dataset" and this table contains a key field that holds the unique dataset name. Another imposed field is an internal status field, which enables the search to hide datasets that are known to be bad. Apart from these constraints, the dataset catalogue schema can differ from one another. A catalogue of Monte-Carlo datasets has a different set of attributes than that of a catalogue of datasets coming from the detector data acquisition.

Some global catalogues are maintained. The largest one is a list of all datasets known to AMI. It guarantees that the dataset names are unique across all catalogues. The dataset provenance information is also managed globally by using a database representation of a direct acyclic graph. The third type of global table is for use in the manner of reference tables. These tables constrain the values which can be used for certain parameters.

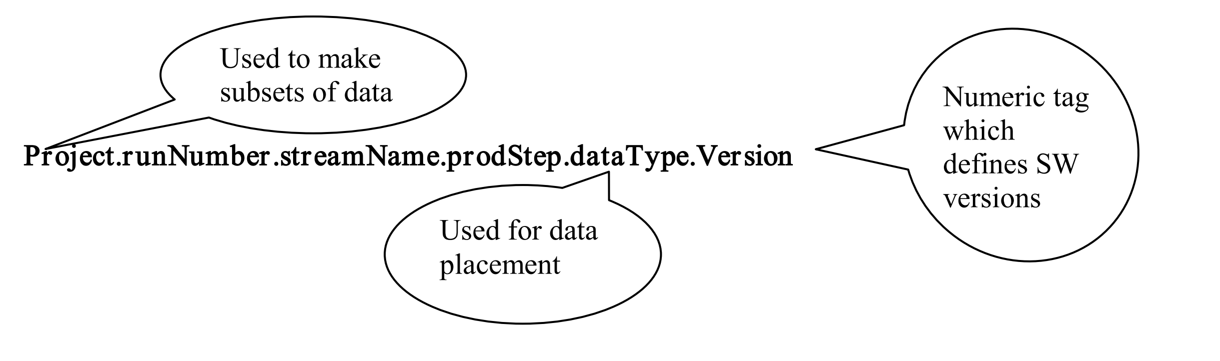

Used to make subsets of data
Project.runNumber.streamName.prodStep.dataType.Version
Numeric tag which defines SW versions
Used for data placement

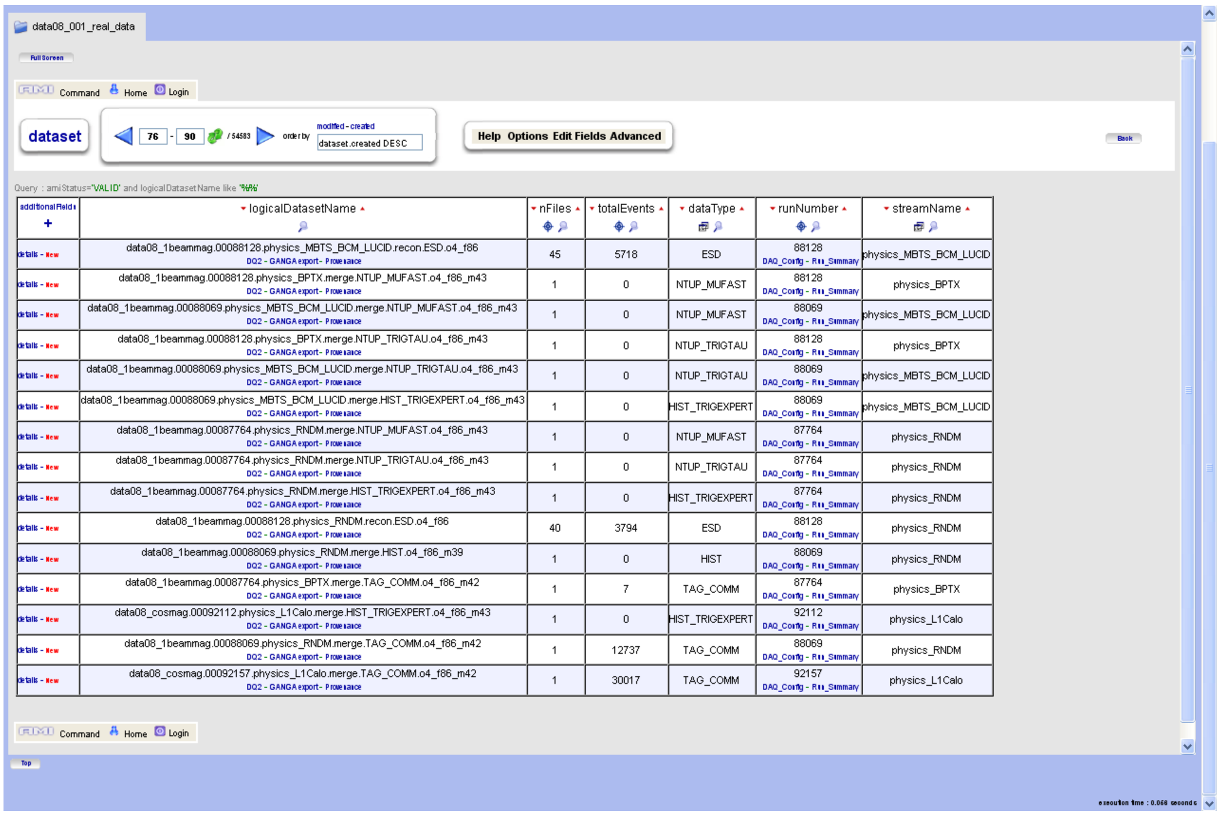
Figure 3 : The nomenclature template for a primary real data set; examples of constrained fields are given.

In 2008 ATLAS introduced a stricter control of nomenclature for datasets [8]. Figure 3 gives an example of the nomenclature for one type of dataset. This dataset nomenclature is quite mnemonic; dataset names have a fixed set of fields, and the values of most of these fields are restricted. Functions to manage the nomenclature have been implemented in AMI [9] using the global reference tables. Coordinators can define new allowed values for the fields using AMI, and the information is circulated automatically by email to the groups that need it. For instance the distributed data management organizes data storage according to the data type and project name.

Collaborating applications that need to name datasets have direct read access on the AMI reference tables so it is possible for them to control nomenclature.

4.2. Data sources

AMI extracts and correlates the information which users need for dataset selection from the primary sources of that information. This extraction, or data pulling, is done by a special AMI task server. Tasks are run as JAVA threads, and scheduled with variable frequency using a task control database table.



dataset.created DESC

Query : amiStatus='VALID' and logicalDatasetName like '%%'

additionalFields +	logicalDatasetName	nFiles	totalEvents	dataType	runNumber	streamName
details - New	data08_1beammag.00088128.physics_MBTS_BCM_LUCID.recon.ESD.o4_f86 DQ2 - GANGA export - Provenance	45	5718	ESD	88128 DAQ_Config - Run_Summary	physics_MBTS_BCM_LUCID
details - New	data08_1beammag.00088128.physics_BPTX.merge.NTUP_MUFAST.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_MUFAST	88128 DAQ_Config - Run_Summary	physics_BPTX
details - New	data08_1beammag.00088069.physics_MBTS_BCM_LUCID.merge.NTUP_MUFAST.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_MUFAST	88069 DAQ_Config - Run_Summary	physics_MBTS_BCM_LUCID
details - New	data08_1beammag.00088128.physics_BPTX.merge.NTUP_TRIGTAU.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_TRIGTAU	88128 DAQ_Config - Run_Summary	physics_BPTX
details - New	data08_1beammag.00088069.physics_MBTS_BCM_LUCID.merge.NTUP_TRIGTAU.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_TRIGTAU	88069 DAQ_Config - Run_Summary	physics_MBTS_BCM_LUCID
details - New	data08_1beammag.00088069.physics_MBTS_BCM_LUCID.merge.HIST_TRIGEXPERT.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	HIST_TRIGEXPERT	88069 DAQ_Config - Run_Summary	physics_MBTS_BCM_LUCID
details - New	data08_1beammag.00087764.physics_RNDM.merge.NTUP_MUFAST.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_MUFAST	87764 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_1beammag.00087764.physics_RNDM.merge.NTUP_TRIGTAU.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	NTUP_TRIGTAU	87764 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_1beammag.00087764.physics_RNDM.merge.HIST_TRIGEXPERT.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	HIST_TRIGEXPERT	87764 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_1beammag.00088128.physics_RNDM.recon.ESD.o4_f86 DQ2 - GANGA export - Provenance	40	3794	ESD	88128 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_1beammag.00088069.physics_RNDM.merge.HIST.o4_f86_m39 DQ2 - GANGA export - Provenance	1	0	HIST	88069 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_1beammag.00087764.physics_BPTX.merge.TAG_COMM.o4_f86_m42 DQ2 - GANGA export - Provenance	1	7	TAG_COMM	87764 DAQ_Config - Run_Summary	physics_BPTX
details - New	data08_cosmag.00092112.physics_L1Calo.merge.HIST_TRIGEXPERT.o4_f86_m43 DQ2 - GANGA export - Provenance	1	0	HIST_TRIGEXPERT	92112 DAQ_Config - Run_Summary	physics_L1Calo
details - New	data08_1beammag.00088069.physics_RNDM.merge.TAG_COMM.o4_f86_m42 DQ2 - GANGA export - Provenance	1	12737	TAG_COMM	88069 DAQ_Config - Run_Summary	physics_RNDM
details - New	data08_cosmag.00092157.physics_L1Calo.merge.TAG_COMM.o4_f86_m42 DQ2 - GANGA export - Provenance	1	30017	TAG_COMM	92157 DAQ_Config - Run_Summary	physics_L1Calo

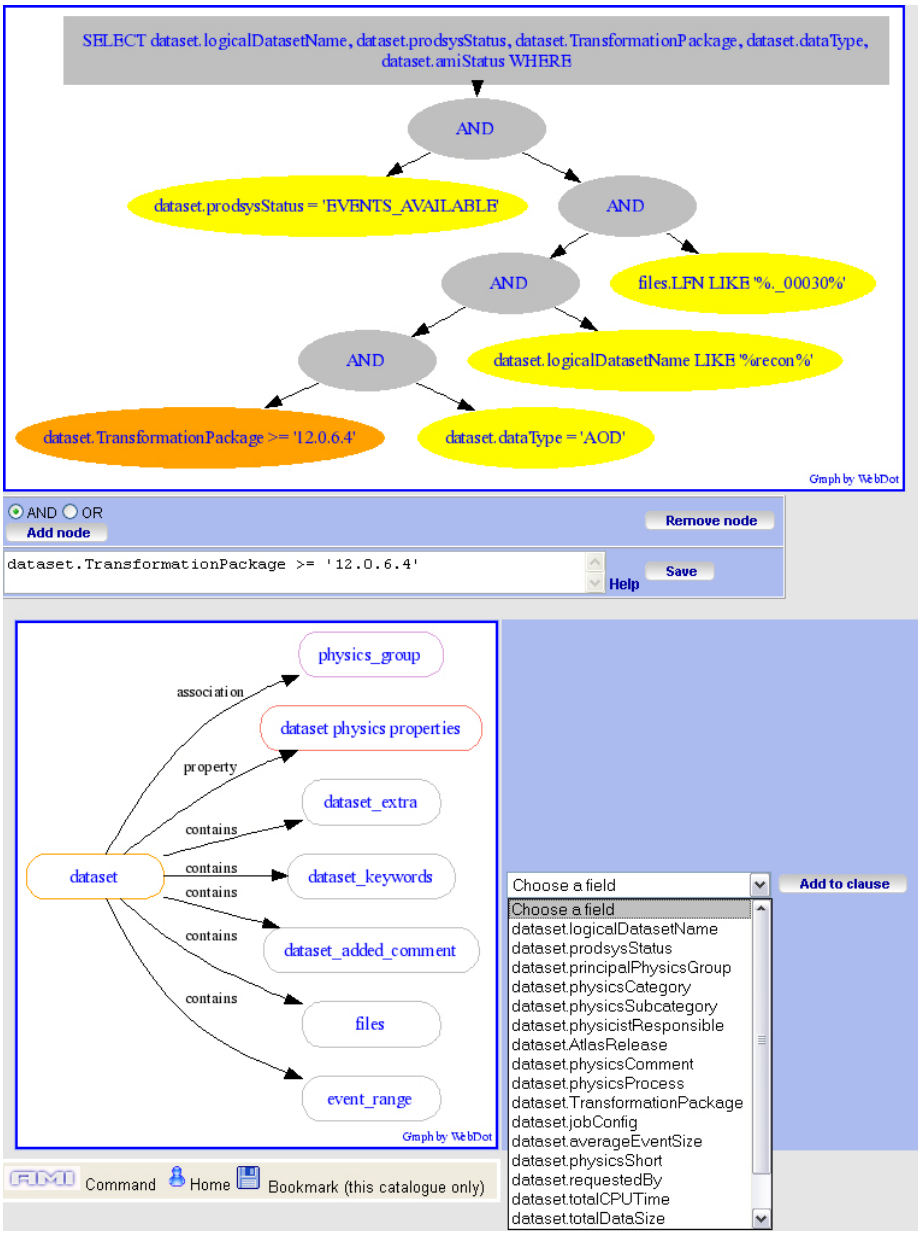
Figure 4 : Result of a query. Links to other applications are shown in dark blue.

The current sources for dataset information stored in AMI are the Tier 0 management system [10], the ATLAS production database [11], the production Task Request system [12], the ATLAS SVN repository, and other AMI based applications such as the Tag Collector [13].

Datasets formed at Tier 0 appear in AMI within 5 minutes. State changes of simulated datasets are also seen within about 5 minutes, but their treatment may take much longer, depending on the number of files produced by a simulation task. Less essential updates like calculating the links to event generation job options files in the SVN repository are done at a much slower cadence, typically every 6 hours.

It is also possible for authorized physicists to update information in the dataset catalogues using either the web interface or a web service client. For example the Monte Carlo coordinator can correct cross-section values that are in some cases reported incorrectly in generator log files.

Of course we do not aim to copy all the metadata available from our data sources. It is essential only to have a handle on the data in other metadata databases. For example AMI keeps a record of the production system job execution ID for each file, and this can be used to obtain the complete job record.



```
SELECT dataset.logicalDatasetName, dataset.prodsysStatus, dataset.TransformationPackage, dataset.dataType,
dataset.amiStatus WHERE
```

```
dataset.TransformationPackage >= '12.0.6.4'
```

Figure 5 : A screen shot of the AMI interface for refining a standard query. The user sees a graphical representation of the gLite query, which he or she can modify by adding a condition on any field in the schema.

The AMI web interface can link to information in another ATLAS metadata application by either providing the URL to the application's own web site or by considering the other application to be a part of AMI. The latter functionality is implemented by entering an AMI compatible description of the external database in the AMI router database. All the AMI generic software will then work on the external database because it is seen as part of AMI. In this way we can provide web browsers for ATLAS databases that do not have their own. The only requirement is that the database tables thus described have single column primary and foreign keys and, of course, we must have a read access to the database. Some typical search results are shown in Figure 4.

4.3. Fixed queries on AMI catalogues
The most common query on AMI starts with a part of the dataset name. It is also possible to search using a subset of parameters, such as projectName and SW Release version, or a search based on keywords attached to datasets. The simplest dataset search gives access to these 'fixed" queries, generated in gLite grammar and sent to all AMI dataset schema. Once a reply has been obtained (see Figure 4) users may use the results to navigate to more details, change the look of the display, or modify the query itself, as described in the next section.

4.4. Arbitrary queries on AMI catalogues
Because of the generic, multi-catalogue structure of AMI the parameters that the user can define in the "fixed" query interfaces are those which are present in the majority of catalogues. But we also provide a means to build an arbitrary SQL request on a selected catalogue. A graphical interface shows the user the complete schema for the selected catalogue, and guides the selection of the fields. As an example, one set of ATLAS users was interested only in those datasets of a catalogue which have the AOD format and which have exactly 30 files. A query constructed using the "Refine Query" function permits this list to be provided. This interface is shown in Figure 5; it is constructed dynamically by introspection of the schema.

5. The Web Service
AMI has provided a classic "SOAP" web interface since 2003. The AMI API now contains over 50 different commands for different actions on the dataset catalogues, although in many cases they are just wrappers to generic commands. We decided not to represent these commands individually in the web service interface (the "WSDL" file) but to keep the client very simple. The client process can be summarized as "send us AMI a command; get the reply". The command and its arguments are sent to the server, the command is executed on the server, and then the XML of the reply can be used directly by the client, or transformed using one of the XSLT files we provide. The advantage of this approach is that we do not have to modify the client in any way if a new command is added on the server. The disadvantage is that clients must rely on independent documentation to know which commands are available.

The web interface implementation uses the same set of commands as those that are available for the web service, but in this context the XML output of the command is transformed to HTML. The web interface provides a means to show the AMI command which was actually used. So it is possible to extract the command on the web, recuperate the syntax and then paste it into a script for execution via the web service.

The web interface also provides a special "web service" page where users can try out web service commands, without having to install a client.

One of the advantages of a web service is that in theory a client can be generated for any modern computing language. But we do not expect the average physicist to be able or willing to do this. An AMI client written in PYTHON (pyAMI) is available in the ATLAS software release. It will return results either as a PYTHON object or in text. We have provided PYTHON wrappers of many commands and in some cases have aggregated the basic commands to provide more complex functions. For instance the wrapper "amiGetDatasetEVNTInfo" takes an arbitrary Monte Carlo dataset name as its argument, and returns selected information about the parent event generation data

A SOAP interface is not well adapted for all client needs, and so we have also provided a Representational State Transfer (REST) type interface for some special functions.

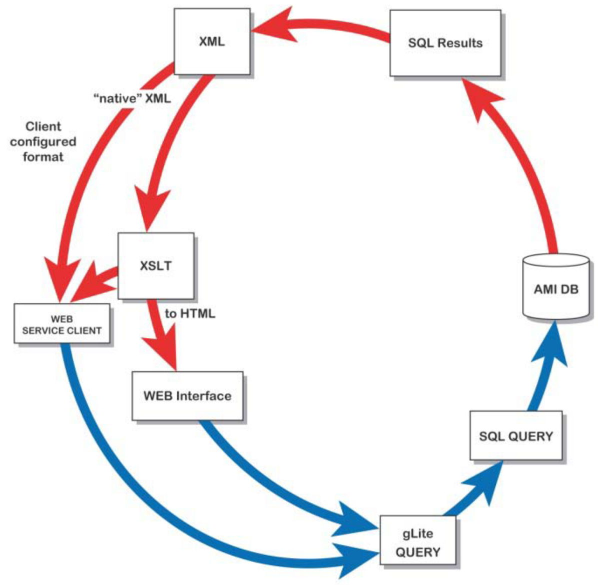


Figure 6 : The XML/XSLT mechanism and the use of gLite grammar.

6. Current status and prospective

6.1. Quality of service

In 2008 AMI and Tag Collector were moved from Grenoble to the French Tier 1 site at Lyon, whose staff now manages all the infrastructure tasks of these applications. The dataset catalogues are hosted on ORACLE, using 4 ATLAS dedicated instances; Linux RHEL 5.2 64 Bits (bi-pro quadri-core) machines, ORACLE version 10.2.0.4. The Atlas Release databases for Tag Collector use mySQL, running at Lyon. The AMI /Tag Collector services are running on two servers, using the TOMCAT web container. A front end Apache server manages the load balancing. User sessions are routed transparently to one of the two servers.

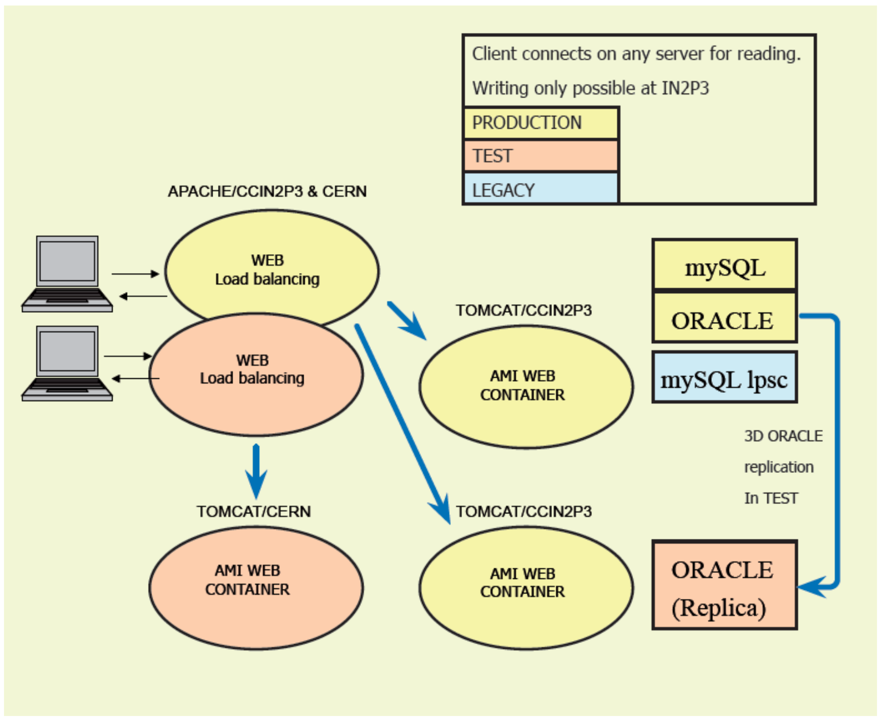


Figure 7 Deployment of AMI and Replication

These applications are unusual for ATLAS because most essential services run at CERN. In order to increase the availability of the dataset search, the AMI dataset catalogues are replicated to ORACLE at CERN using the 3D mechanism [14] which is based on ORACLE streams.

This means that users who are interested only in reading AMI can do so using the CERN replica, if they wish. If the ORACLE database at Lyon is unavailable for any reason, the AMI web interface will redirect users requesting read-only operations to the AMI replica at CERN. All writing operations are done on the master database. For various reasons, the streaming mechanism is not the method of choice for ensuring redundancy. Nevertheless, if there were to be a serious problem at the CCIN2P3 it would be possible to change the CERN replica to the master in a few hours.

All the other replicated database applications are distributed FROM CERN to Tier 1 sites.

6.2. Performance and Scalability aspects of AMI
A well known problem with database applications which aim to support several database back ends is that they cannot easily optimise queries for a particular database engine. To some extent we have been able to counter this problem by a polymorphic design of connection software. At the lowest level of the software (See Figure 1) the database connections “know” to which engine they are connected. We use this level to manage specific RDBMS syntax and frequent non-application specific database operations can be optimized therein. A second application strategy is to remove very time critical operations to the top application level, and allow them to by-pass the generic query mechanism. When AMI is reading the Tier 0 database to get the information about the latest ATLAS datasets, there is no reason to check that the AMI catalogue receiving the data has a schema which is correct for ATLAS real datasets, performed in the AMI generic software layers, thus one can exploit bulk inserts and bind variables at this level.

Our use of threaded generic queries to independent ORACLE schema has turned out to be very successful, we are sure it will be scalable as it implies that we can easily archive old data and as yet we have had no need for the possibility ORACLE gives of partitioning large tables; if we were to do this it would be transparent to the AMI software. The same is true of the creation of any special database specific kind of index.

A typical successful dataset search returns in under 2 seconds on the web interface. The more complex "keyword" search, over many fields and tables, returns in about 10 seconds.

Over the past 12 months the number of web site visitors has increased by 60%, and we typically support at least 50 TOMCAT sessions on each of our servers, peaking to several hundred when the nightly build mechanism is reading from the Tag Collector database. (Our servers do not distinguish between the two applications.)

Although our current performance is more than acceptable we are aware that several things could be done to improve performance. However our first priority is to increase the functions available.

6.3. Conclusions
AMI has shown itself to be a powerful tool which can be used for the physics metadata catalogues of ATLAS. Both datasets and files will be catalogued, from real and from simulated data. Searching is fast and flexible. Metadata is correlated from several input sources and held in the AMI tables. The AMI interface is one of the main entry points for all ATLAS metadata sources.

ATLAS metadata information is available from many sources, with different granularities. Users need to be able to navigate from one granularity to another in order to find and combine the data from these different sources. This is not a trivial task since the information is distributed across many applications, and each source application presents a different interface and exports the data in a different format. A mediator interface is defined as “a software module that exploits encoded knowledge about some sets or subsets of data to create information for a higher layer of applications” [15]. To put it another way, a mediator is an application which puts some domain expertise between the user and a group of data sources, so that information coming from these different sources can be aggregated. Our ambition is that AMI will become a "mediator" for ATLAS metadata.

References

[1] http://ami.in2p3.fr
[2] S. Albrand, T. Doherty, J Fulachier and F. Lambert;The ATLAS metadata interface; *J. Phys. Conf. Ser.* ***119*** Volume 119 (2008) 072003 (10pp)
[3] Elizabeth Gallas, David Malon, Solveig Albrand and Eric Torrence; An Overview of Metadata in ATLAS. *(this conference)*
[4] http://edg-wp2.web.cern.ch/edg-wp2/security/voms/
[5] Thomas Doherty, http://ppewww.physics.gla.ac.uk/preprints/2007/01/
[6] Peter Kunszt and Ricardo Rocha https://edms.cern.ch/file/573725/1.2/EGEE-TECH-573725-v1.2.pdf
[7] Birger Koblitz and Nuno Santos The gLite AMGA Metadata Catalogue ;2006 *Proc. Int. Conf . on Computing in High Energy and Nuclear Physics (CHEP-06)* vol II (Macmillan India) p 820
[8] S. Albrand et al. ATLAS Internal Report http://cdsweb.cern.ch/record-restricted/1070318/
[9] http://ami.in2p3.fr/opencms/opencms/AMI/www/ReferenceTables/
[10] Armin Nairz and Luc Goossens, The ATLAS Tier-0: Overview and Operational Experience, *(this conference)*
[11] https://twiki.cern.ch/twiki/bin/view/Atlas/ProdSys
[12] Pavel Nevski, Alexei Klimentov and Alexandre Vaniachine, . Knowledge Management System for ATLAS Scalable Task Processing on the Grid; *(this conference)*
[13] Emil Obreshkov et al. Organization and Management of ATLAS Software Releases; *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Volume 584, Issue 1, 1 January 2008, Pages 244-251
[14] Maria Girone. Distributed Database Services - a Fundamental Component of the WLCG Service. *(this conference)*
[15] Gio Wiederhold, Mediators in the architecture of future information systems, *IEEE Computer Magazine*, Vol 25, No3, p38-49 March 1993