

Using the AMGA Metadata Catalog

Last review date	Reviewer
2009-09-15	Marco Bencivenni Enrico Fattibene

Table of Contents

[Using the AMGA Metadata Catalog](#)

[Introduction](#)

[AMGA Metadata Basic Concepts](#)

[Accessing AMGA from the command line](#)

[Basic commands](#)

[mdcli/ mdclient general commands](#)

[Handling schemas and attributes](#)

[Handling entries and metadata](#)

[Querying metadata](#)

[More documentation](#)

Using the AMGA Metadata Catalog

This document has the goal of providing basic information on the usage of the AMGA Metadata Catalog.

Introduction

The *AMGA Metadata Catalog* is the EGEE *gLite* service that allows metadata handling on the grid. The main usage can be as a "Front-end" file metadata service, providing means of describing and discovering data files required by users and their jobs. It can also be used as a Grid- Enabled Database for applications that require to structure their data, providing a database- like service supporting Grid Security features (X509 Proxies and the VOMS authentication and authorization system). Finally, an additional feature allows the accessing of existing relational databases from a grid environment (*Worker Nodes*, *User Interface*, etc), which enables the addition of Grid Security to existing DBs.

Users and applications can interact with an AMGA server using *command line tools*:

- `mdcli/ mdclient` available for Scientific Linux flavours (they can be built easily to other platforms)
- `mdjavacli/ mdjavaclient`, the Java versions of the previous one that allows the interaction from any platform

or through *APIs*, available for:

- C++
- Java
- Python
- Perl
- PHP

Some of the `mdcli/ mdclient` command line tools are explained below.

AMGA Metadata Basic Concepts

There are certain fundamental concepts which must be understood when dealing with AMGA

- **Entry** - it is the representation of the real world entity which we are attaching metadata to in order to describe it
- **Attribute** - key/ value pair. It has:
 - *Type* - The type (int, float, string, etc...)
 - *Name/ Key* - The name of the attribute
 - *Value* - The value of an entry's attribute
- **Schema** - A set of attributes
- **Collection** - A set of entries associated with a schema
- **Metadata** - The list of attributes (including their values) associated with entries

If we want to make an analogy with the RDBMS world, we have the following:

- schema = table schema
- collection = database table
- attribute = schema column
- entry = table row/ record

By analogy with a file system, we have:

- collection = directory
- entry = file

In the AMGA help and documentation, often directory is used to refer to collection, as file refers to entry

Example: Metadata for movies

Movie files (*entries*) could be saved on *Grid Storage Elements* and registered into a *File Catalogue*. We want to add *metadata* to describe the movie content. A possible *schema* could be:

- `Title -- varchar`
- `Runtime -- int`
- `Cast -- varchar`
- `LFN -- varchar`

We can use the GUID of the file as the *entry* name.

A *collection* named `movies` of an AMGA server could be the repository of the movies' metadata and will allow to find the movies satisfying users' queries.

Accessing AMGA from the command line

To start using AMGA from the command line, you need to use either `mdcli` or `mdclient` executables. They have to be installed, together with the required libraries, into a *User Interface* (and on the *Worker Nodes* of the sites where you plan to run jobs that will access AMGA). These do not come by default within the standard `gLite UserInterface` and `WorkerNodes` packages (on `gLite 3.1` they should be available), so you need to install them manually.

You can download RPMs for SLC here:

- <http://amga.web.cern.ch/amga/downloads/glite-amga-cli-1.3.0-1.SLC4.i386.rpm> for SLC4 systems
- <http://amga.web.cern.ch/amga/downloads/glite-amga-cli-1.3.0-1.i386.rpm> for SLC3 systems

Once installed, you need to properly configure the configuration file: `$HOME/.mdclient.config`. A template of this file can be found in `$GLITE_INSTALLATION/etc/mdclient.config`. This also behaves as a system wide configuration file, useful in a multi- user system (like on *Worker Nodes*) and it will be read by the AMGA clients if the `$HOME/.mdclient.config` does not exist.

The relevant values of the `mdclient.config` are the following:

```
Host = amga.ct.infn.it
Port = 8822
Login = NULL
PermissionMask = rwx
GroupMask = r-x
Home = /gilda
UseSSL = yes
AuthenticateWithCertificate = 1
UseGridProxy = 1
VerifyServerCert = 0
```

where

- *Host* defines the AMGA server you want to use. You can use `amga.ct.infn.it` for the purpose of testing (installed as part of the EGEE GILDA training infrastructure)
- *Port* defines the port where the AMGA server daemon is listening. It should not be changed
- *Login* defines the credential to be used to authenticate within the AMGA catalog. You can use:
 - your `username`, if you have requested one to AMGA server administrator providing him your Grid Certificate Distinguish Name (DN)
 - or use `NULL`, to be authenticated as a generic VO user (given the fact that your VO is supported by the AMGA server you are going to use)
- The AMGA server in the GILDA training infrastructure supports the following VOs: `gilda`, `eela`, `eumed`, `euchina`, `cometa`. Please contact the `amga.ct.infn.it` sysadmins (`tony.calanducci[at]ct.infn.it`) to request support for your VO or to get a personal account. Soon a registration page will be available at: <https://amga.ct.infn.it:8443/register>
- Please take a look at the AMGA documentation [here](#) for all the available options of the `.mdclient.config` file.
- The options on the previous example were set up to be authenticated as a generic VO user using your Grid Proxy. Be sure to initialize your proxy with the proper command:

```
-bash-2.05b$ voms-proxy-init --voms gilda
Enter GRID pass phrase:
Your identity: /C=IT/O=GILDA/OU=Personal Certificate/L=INFN Catania/\
CN=Tony Calanducci/Email=tony.calanducci@ct.infn.it
Creating temporary proxy ..... Done
Contacting voms.ct.infn.it:15001 [/C=IT/O=INFN/OU=Host/L=Catania/CN=voms.ct.infn.it] "gilda" Done
Creating proxy ..... Done
Your proxy is valid until Sun Feb  3 08:04:46 2008
```

Once everything has been set up properly, start the AMGA `mdclient` :

```
-bash-2.05b$ mdclient
Connecting to amga.ct.infn.it:8822...
ARDA Metadata Server 1.3.0
Query>
```

This is an interactive shell similar to `mysql` or `psql` that allows to issue AMGA commands to the server. The other AMGA executable that comes with the RPM, `mdcli`, provides the very same functionality, but it executes ONLY one command at time, passed as command line parameter, and immediately exits. This is quite useful in bash script to be run on Worker Nodes that needs to interact with the metadata catalog.

Basic commands

It is possible to get help anytime on the client just using the `'help'` command.

Try the `help` command

```
Query> help
>> help [topic]
>> Displays help on a command or a topic.
>> Valid topics are: help metadata metadata-optional directory replication constraints \
entry group acl index schema sequence user view site replicas ticket /
capabilities commands
Query>
```

Commands are grouped by topic. You can get the list of valid commands for each topic, typing: `help [topic]`

The list of valid topics is:

- help
- metadata
- metadata- optional
- directory
- replication
- entry
- group
- acl
- index

- schema
- sequence
- user
- view
- ticket
- commands

Try the use of `help` command with any topic

Query> help entry

mdclient general commands

The following tables gives a brief description of the general use commands.

<code>createdir path</code> [options]	Create a new <code>collection</code> . It can inherit (using the <code>inherit</code> option) the schema associated to the upper level collection
<code>rm pattern</code>	Remove the entries corresponding to the given pattern
<code>dir collection</code>	List the content (entries, subcollections, sequences, indexes) of the given collection
<code>listentries</code> <code>collection</code>	List the entries only of the given collection
<code>stat pattern</code>	Show the statistics of an entry or collection
<code>chown file owner</code>	Change the ownership of an entry or collection
<code>chmod file rights</code>	Change the access mode of an entry or collection
<code>rmdir collection</code>	Remove a collection
<code>dump collection</code>	Make a recursive dump starting from a given collection, (the default is: '/')
<code>pwd</code>	Prints the current collection
<code>whoami</code>	Prints the current user
<code>cd collection</code>	Change the current collection

General commands examples:

Query> whoami

```
>> gilda
```

Query> pwd

```
>> /
```

Query> cd / gilda

Query> cd tony

Query> pwd

```
>> /gilda/tony/
```

Query> dir

```
>> /gilda/tony/seq2
>> sequence
>> /gilda/tony/seconda
>> collection
>> /gilda/tony/v1
>> collection
>> /gilda/tony/v2
>> collection
>> /gilda/tony/view1
>> view
>> /gilda/tony/aentry
>> entry
>> /gilda/tony/14
>> entry
>> /gilda/tony/15
>> entry
>> /gilda/tony/16
>> entry
>> /gilda/tony/17
>> entry
>> /gilda/tony/18
>> entry
>> /gilda/tony/20
>> entry
```

Query> listentries

```
>> /gilda/tony/aentry
>> /gilda/tony/14
>> /gilda/tony/15
>> /gilda/tony/16
>> /gilda/tony/17
>> /gilda/tony/18
>> /gilda/tony/20
```

Query> cd seconda

Query> dir

```
>> /gilda/tony/seconda/2
>> entry
```

Query> rm *

Query> cd ..

Query> pwd

```
>> /gilda/tony/
```

Query> rmdir seconda

Handling schemas and attributes

Once a *collection* has been created, its *schema* should be defined, adding one or more *attributes*. As illustrated in the basic concept section, each attribute is defined by its *name* and its *type*.

The command to add a new attribute to a collection schema is the following:

addattr dir attribute_name type where:

- `dir` is the collection/ directory you are adding the attribute to. You can use relative or absolute path to refer to it.
- `attribute_name` is the name you want to give to the attribute you are adding
- `type` specifies what kind of values the attribute is able to contain

AMGA valid *attribute types* and their corresponding types used in the internal AMGA back- end are shown in the following table:

AMGA	PostgreSQL	MySQL	Oracle	SQLite	Python
int	integer	int	number(38)	int	int
float	double precision	double precision	float	float	float
varchar(n)	character varying(n)	character varying(n)	varchar2(n)	varchar(n)	string
timestamp	timestamp w/ o TZ	datetime	timestamp(6)	unsupported	time(unsupported)
text	text	text	long	text	string
numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	numeric(p,s)	float

The AMGA server uses internally a relational database to store all the users' metadata. It can use almost any RDBMS that has an ODBC driver. Most of the installations use PostgreSQL and MySQL. If the types indicated in the first column are used to define attributes, metadata can be moved and replicated easily among AMGA servers that use different DB backends. If you don't mind to metadata portability between servers, you can also use all the specific data types of a given DB back- end (we have tried GIS datatypes and Network datatypes of PostgreSQL, for example). To find out which database back- end a given AMGA server is employing, you can use the command `backend`:

```
Query> backend
>> PostgreSQL
```

To remove an attribute from a collection schema, the following command is used:

```
removeattr dir attribute_name
```

To inspect the schema of a given collection (or of an entry), use:

```
listattr dir/ entry
```

Schema population example:

Let's create a *movies* collection and define its schema, adding the following attributes: *title*, *runtime*, *cast*, *LFN*, *to_remove* (one of them will be removed):

```
Query> createdir /gilda/movies
Query> addattr /gilda/movies title varchar(50)
Query> addattr /gilda/movies runtime int
Query> addattr /gilda/movies cast text
Query> addattr /gilda/movies LFN varchar
Query> addattr /gilda/movies to_remove float
Query> listattr /gilda/movies
>> title
>> varchar(50)
>> runtime
>> int
>> cast
>> text
>> LFN
>> varchar
>> to_remove
>> float
Query> cd /gilda/movies
```

```

Query> removeattr . to_remove
Query> listattr .
>> title
>> varchar(50)
>> runtime
>> int
>> cast
>> text
>> LFN
>> varchar

```

Handling entries and metadata

Once the schema of a collection has been defined, it is possible to add new *entries*. **Each entry must have an entry name.** You can think of entry names as primary keys of a database table. Entry names are unique. According to your purposes, you could have different options. To mention some examples, GUIDs (Globally Unique Identifiers) could be an option if you are adding metadata to files, the final part of JOB IDs ('/' can't be part of entry names) if you are adding metadata to running jobs, or simply an incremental integer number. You may use any appropriate entry name to better describe your entities. If you want to use an incremental integer as entry name, **AMGA sequences** can be very useful. You can define one or more sequences for a given collection, but those will not generate by themselves new numbers unless you explicitly request it.

```

Query> help sequence
>> sequence_create name dir [increment] [start value]
>> Creates a new sequences with given name in the given directory.
>> sequence_next sequence
>> Gets the next value from a sequence.
>> sequence_remove sequence
>> Deletes a sequence.
Query>

```

Sequence examples

Create a sequence for the *movies* collection and get the next sequence id:

```

Query> pwd
>> /gilda/movies/
Query> sequence_create id /gilda/movies
Query> dir
>> /gilda/movies/id
>> sequence
Query> sequence_next /gilda/movies/id
>> 1
Query> sequence_next /gilda/movies/id
>> 2
Query> sequence_next /gilda/movies/id
>> 3

```

Once decided how to handle entry names, we can actually start **adding or removing entries**. Four commands are available for that purpose:

addentries entry1...	Adds one or more entries (also across collections)
addentry entry [attribute_name value]...	Adds one new entry and initializes one or more attributes
removeentries entry1...	Removes one or more entries (also across collections)
rm [- rf] pattern [condition]	Removes entries matching pattern/ condition

Entry creation and deletion examples

Let's add 2 entries with valid attributes and 3 empty entries, then delete the last two:

```

Query> pwd
>> /gilda/movies/
Query> sequence_next id
>> 4
Query> addentry 4 title 'Spiderman 3' runtime 120 cast 'Kirsten Dunst, Tobey Maguire' \
LFN 'lfn:/grid/gilda/movies/spiderman.mov'
Query> sequence_next id
>> 5
Query> addentry 5 title 'Pretty Woman' runtime 95 cast 'Julia Roberts, Richard Gere' \
LFN 'lfn:/grid/gilda/movies/prettywoman.mov'
Query> sequence_next id
>> 6
Query> addentries 6 7 8
Query> dir
>> /gilda/movies/id
>> sequence
>> /gilda/movies/4
>> entry
>> /gilda/movies/5
>> entry
>> /gilda/movies/6
>> entry
>> /gilda/movies/7
>> entry
>> /gilda/movies/8
>> entry
Query> removeentries 7 8
Query> dir
>> /gilda/movies/id
>> sequence
>> /gilda/movies/4
>> entry
>> /gilda/movies/5
>> entry
>> /gilda/movies/6
>> entry

```

addentries entry1...	Adds one or more entries (also across collections)
addentry entry [attribute_name value]...	Adds one new entry and initializes one or more attributes
removeentries entry1...	Removes one or more entries (also across collections)
rm [- rf] pattern [condition]	Removes entries matching pattern/ condition

There are three more useful commands for handling the value of attributes:

getattr pattern attribute1 attribute2 ...	Returns the values of the attributes for all files matching pattern
setattr entry attribute value [attribute value] ...	Sets given attributes to specified values for all entries matching entry
clearattr entry attribute	Sets the attribute to NULL for all entries matching entry pattern.

Let's use the previous command to set and get entry attributes'values:

```

Query> pwd
>> /gilda/movies/
Query> getattr * title
>> 4
>> Spiderman 3
>> 5
>> Pretty Woman
>> 6
>>
Query> getattr 6 title
>> 6
>>
Query> setattr 6 title 'Armageddon'
Query> setattr 6 runtime 150 cast 'Bruce Willis, Ben Affleck' \
LFN 'lfn:/grid/gilda/movies/armageddon.mov'
Query> getattr /gilda/movies/ title LFN cast
>> 4
>> Spiderman 3
>> lfn:/grid/gilda/movies/spiderman.mov
>> Kirsten Dunst, Tobey Maguire
>> 5
>> Pretty Woman
>> lfn:/grid/gilda/movies/prettywoman.mov
>> Julia Roberts, Richard Gere
>> 6
>> Armageddon
>> lfn:/grid/gilda/movies/armageddon.mov
>> Bruce Willis, Ben Affleck

```

Querying metadata

Finally, after we have created a collection, defined its schema, added entries with their attribute values to it, we can issue a query to get back the information we need.

The most used command to issue queries is `selectattr`. Its syntax is as follows:

```
selectattr collection_name:attribute_name... condition
```

which returns the values of given attributes for all files matching the condition where:

- `collection_name` specifies the path of the attribute's collection we want to print out. If it's in current collection, the `'.'` (dot) is mandatory. If more than one attribute will follow and they are in the same collection of the first one, the `collection_name` can be omitted
- `attribute_name` specifies the attribute whose values we want to print out
- `condition` specifies a condition on attributes to filter the result set. Logical (and/ or/ not), comparison, aggregation operators can be used. Joins (inner, outer, left, right) between schemas are allowed. `Limit,order,distinct,group` are also available. [Here](#) you can find a summary of all the available operators and options. If you don't want to give any condition, use a pair of `"` (single quotes).

A simpler query command is `find`:

```
find pattern condition
```

It returns only the names of the entries that match the pattern and satisfy the condition.

Some examples of queries:

To print the titles and the LFNs of all the movies whose runtime is greater than 80 minutes:

```
Query> selectattr /gilda/movies:title LFN 'runtime > 80'  
>> Spiderman 3  
>> lfn:/grid/gilda/movies/spiderman.mov  
>> Pretty Woman  
>> lfn:/grid/gilda/movies/prettywoman.mov  
>> Armageddon  
>> lfn:/grid/gilda/movies/armageddon.mov
```

To print the titles and the runtime of the movies where Julia Roberts performed:

```
Query> pwd  
>> /  
Query> cd /gilda/movies  
Query> pwd  
>> /gilda/movies/  
Query> selectattr .:title runtime 'like(cast, "Julia%")'  
>> Pretty Woman  
>> 95
```

To issue the last query example using find:

```
Query> find /gilda/movies/ 'like(cast, "Julia%")'  
>> 5  
Query> getattr 5 title runtime  
>> 5  
>> Pretty Woman  
>> 95
```

More documentation

- [AMGA homepage](#)
- [AMGA User's and Administrator's Manual\(PDF\)](#)
- [From the GILDA Twiki: Metadata - Introduction to AMGA](#)
- [From the GILDA Twiki: AMGA Advanced usage](#)
- [From the GILDA Twiki: Accessing pre- existing databases through AMGA](#)