Deploying and administrating the ATLAS Metadata Interface (AMI) 2.0 ecosystem

Jérôme Odier[1,*], *Jérôme* Fulachier[1], *Fabian* Lambert[1]

[1]Laboratoire de Physique Subatomique et de Cosmologie, Université Grenoble-Alpes, CNRS / IN2P3, 53 rue des Martyrs, 38026, Grenoble Cedex, FRANCE

Abstract. ATLAS Metadata Interface (AMI) is a generic software ecosystem for metadata aggregation, transformation and cataloguing. Benefiting from about 20 years of feedback in the LHC context, the second major version was released in 2018. This paper describes how to install and administrate AMI version 2. A particular focus is given to the registration of existing databases in AMI, the adding of additional metadata and, finally, the generation of high level HTML 5 search interfaces using a dedicated wizard.

1 Introduction

Originally developed for the ATLAS experiment [1] at the CERN Large Hadron Collider (LHC), ATLAS Metadata Interface (AMI) is a generic ecosystem for metadata aggregation, transformation and database storing. Benefiting from about 20 years of feedback [2, 3], it provides a wide array of tools (command line tools, lightweight clients) and Web interfaces for searching data by metadata criteria.

The second version of AMI, released in 2018, was designed to guarantee scalability, evolutivity and maintainability. It perfectly fits the needs of scientific experiments in big data contexts. The design principles and main features were described in [4]. A key feature is the AMI Metadata Query Language (MQL) [5], a Domain-Specific Language (DSL) for querying databases and permitting to perform queries without (precisely) knowing relations between tables. In other words, it means that MQL only deals with metadata names while SQL uses a catalog / table / field paradigm.

This paper describes how to install and administrate the AMI ecosystem. The following sections present the hardware and software requirements, the deployment and the administration of both the Java backend and the JavaScript frontend of AMI. And finally, a particular focus is given to the registration of existing databases in AMI and to the graphical design of rich HTML 5 search interfaces with a dedicated wizard.

* jerome.odier@lpsc.in2p3

2 Installing AMI

Installing the AMI ecosystem is quite easy and doesn't require any specialist skills (system, database, development, …). Most operations are automated and a set of HTML 5 interfaces makes the ecosystem administration easy.

2.1 Hardware and software requirements

Any machine (x68, x86-64, ARM, …) with at least 4 gigabytes RAM is able to run the AMI ecosystem. It can be deployed on Linux, Mac OSX and Microsoft Windows. Table 1 shows the minimum software requirements for each software of the ecosystem.

Table 1. Minimum software requirements for both deployment and compilation from sources.

- for deployment -			
AMI Java Core	**AMI Web Server**	**AMI Task Server, see [4]**	**AMI Web Framework**
Oracle Java 11 or OpenJDK 11	Oracle Java 11 or OpenJDK 11, Apache Tomcat 9	Oracle Java 11, Apache Maven 3	Python 2 or 3

- for compilation -			
AMI Java Core	**AMI Web Server**	**AMI Task Server, see [4]**	**AMI Web Framework**
Oracle Java 11 or OpenJDK 11, Apache Maven 3	Oracle Java 11 or OpenJDK 11, Apache Maven 3, Apache Tomcat 9	Oracle Java 11, Apache Maven 3	Python 2 or 3, Node.js 12

The AMI ecosystem stores its configuration in a dedicated database: the configuration database (also called "router" database for historical reasons). Table 2 shows the list of supported Database Management System (DBMS).

Table 2. Supported DBMS by the AMI ecosystem for the AMI configuration database (at left) and for storing metadata (at right).

- **supported Database Management System -**	
For the AMI configuration database	**For storing metadata in AMI**
MySQL 5, MariaDB 5, PostgreSQL 9, Oracle 12 (11 with restrictions), SQLite 3, H2 1.4	The ones in the left column, Neo4J 3, most of SQL or NoSQL DSMS with a Java Database Connectivity (JDBC [6]) support => after development of an AMI encapsulation class

2.2 Installing and configuring Apache Tomcat 9

In the following, it is assumed that AMI is deployed on Linux in `/opt`. The first step is to install Oracle Java / OpenJDK 11 or above [7] and Apache Tomcat 9 or above [8], the open-source implementation of the Java Servlet [9].

```
> cd /opt
> tar xzf openjdk-13.0.2_linux-x64_bin.tar.gz
> tar xzf apache-tomcat-9.0.30.tar.gz
> rm -fr apache-tomcat-9.0.26/subapps/*
```

The second step is to properly specify the Java options in `/opt/apache-tomcat-9.0.26/bin/setenv.sh` (max memory, 2GB, and stack, 20MB, can be increased if needed):

```
JAVA_HOME=/opt/jdk-13.0.2

JAVA_OPTS='-Djava.awt.headless=true -Djava.security.egd=file:/dev/./urandom'

CATALINA_OPTS='-Xms1G -Xmx2G -Xss20m -server -XX:+UseParallelGC'
```

Finally, it is highly recommended to enable a Hypertext Transfer Protocol Secure (HTTPS) connector on the 8443 TCP port in order to increase security and to enable X.509 certificate authentication. The following XML fragment has to be inserted in `/opt/apache-tomcat-9.0.26/conf/server.xml` (see https://tomcat.apache.org/tomcat-9.0-doc/):

```
<Connector port="8443"
           protocol="org.apache.coyote.http11.Http11NioProtocol"
           connectionTimeout="20000"
           maxThreads="200"
           packetSize="65536"
           compression="on"
           SSLEnabled="true"
           scheme="https"
           secure="true">

    <SSLHostConfig certificateVerification="want"
                   truststoreFile="myTruststoreFile.jks"
                   truststorePassword="myTruststorePassword">

        <Certificate certificateKeystoreFile="myKeystoreFile.jks"
                     certificateKeystorePassword="myKeystorePassword" />

    </SSLHostConfig>

</Connector>
```

2.3 Deploying the AMI Web Server

Deploying AMI in an Apache Tomcat server is completely trivial. The last compiled package is available on GitHub:

```
> curl -L https://raw.githubusercontent.com/ami-
team/AMICore/master/dist/AMI.war > /opt/apache-tomcat-9.0.26/webapps/AMI.war
```

2.4 Deploying the AMI Web Framework

There are two ways of deploying the AMI Web Framework (AWF): i) On an Apache/ Nginx server; ii) Directly on Apache Tomcat. In this paper, the second way is described:

```
> mkdir /opt/apache-tomcat-9.0.26/webapps/ROOT
> cd /opt/apache-tomcat-9.0.26/webapps/ROOT
> curl -L https://raw.githubusercontent.com/ami-
team/AMIWebFramework/master/tools/awf_stub.py > webapps/ROOT/awf.py
> python3 awf.py --update-prod        # for installing the AMI Web Framework
> python3 awf.py --create-home-page   # for creating “index.html”
Page title:
AMI                                   # user free page title
Service URL:
https://<domain>:8443/AMI/FrontEnd    # give your domain
```

For locally testing, `<domain>` can be `localhost`. The service URL can be modified later, directly in `index.html`.

2.4 Starting and stopping AMI

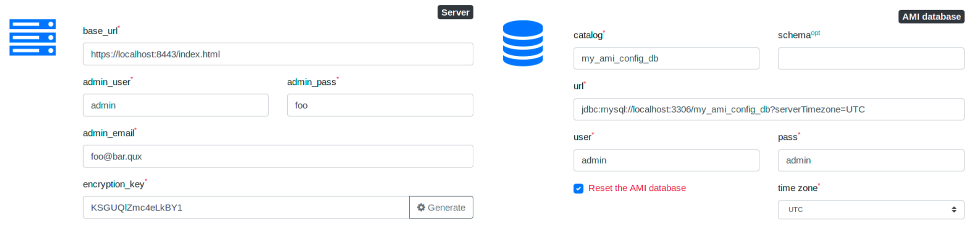
```
> /opt/apache-tomcat-9.0.26/bin/catalina.sh start

> /opt/apache-tomcat-9.0.26/bin/catalina.sh stop
```

3 Configuring AMI

3.1 Finalizing the installation

Before using the ecosystem, the administrator password has to be defined and the AMI configuration database has to be properly created and filled. A dedicated Web interface is devoted to that (URL: https://<domain>:8443/AMI/Setup, see Figure 1).



Server

base_url*	https://localhost:8443/index.html
admin_user*	admin
admin_pass*	foo
admin_email*	foo@bar.qux
encryption_key*	KSGUQlZmc4eLkBY1

AMI database

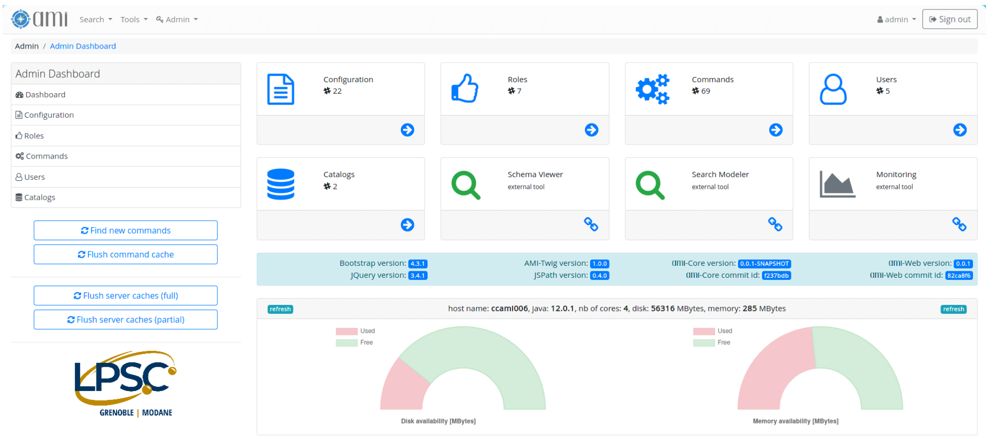
catalog*	my_ami_config_db
schema^opt	
url*	jdbc:mysql://localhost:3306/my_ami_config_db?serverTimezone=UTC
user*	admin
pass*	admin
Reset the AMI database	✓
time zone*	UTC

Fig. 1. Screenshot of the installation finalization interface (URL: https://<domain>:8443/AMI/Setup).

From this point, AMI is accessible from URL: https://<domain>:8443/.

3.2 The administration dashboard

The ecosystem is centrally administrated via a dedicated dashboard. There are five sections: 1) configuration (JDBC connection pool, command cache, logs, authentication modes, user data protection, …); 2) roles; 3) commands; 4) users; 5) catalogs and three external tools (Schema viewer, Search Modeler and Monitoring). The AMI administration dashboard is available from this URL: https://<domain>:8443/?subapp=AdminDashboard (see Figure 3).



Bootstrap version: 4.3.1
JQuery version: 3.4.1
AMI-Twig version: 1.0.0
JSPath version: 0.4.0
ami-Core version: 0.0.1-SNAPSHOT
ami-Core commit id: f237bdb
ami-Web version: 0.0.1
ami-Web commit id: 82ca8f6

host name: **ccami006**, java: **12.0.1**, nb of cores: **4**, disk: **56316** MBytes, memory: **285** MBytes

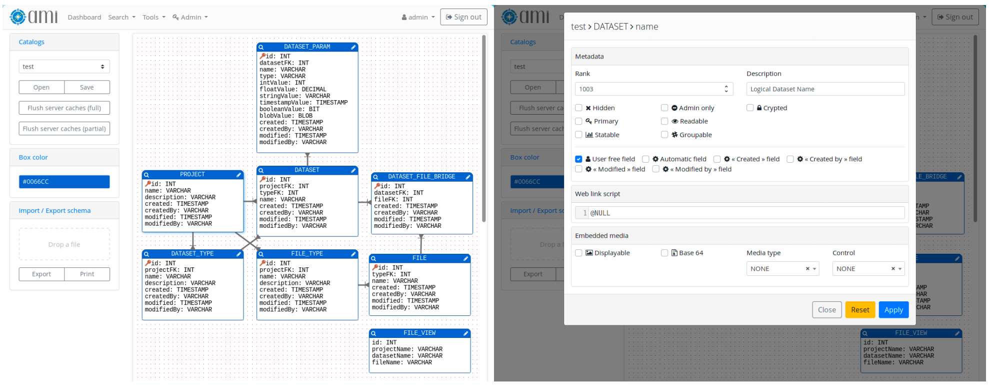
Fig. 2. Screenshot of the AMI centralized administration dashboard (URL: https://<domain>:8443/?subapp=AdminDashboard).

The “catalogs” session permits registering new databases (aka catalogs). It consists of an editable table to specify: catalog name in AMI, JDBC URL, credentials, …

When a new catalog is added, clicking on “Flush Server caches (full)” triggers the extraction of all the catalog, table and field metadata (for instance: field name, type, …), then, it builds a relation graph between tables. This information is stored in a persistent cache and is used by the AMI Metadata Query Language (MQL), as previously said, a query language that automatically builds SQL joins.

3.3 The Schema Viewer

The AMI Schema Viewer is a tool designed for browsing database schemas and for graphically editing additional catalog, table and field metadata. As an example, a field can be tagged as “hidden”, “admin only”, “crypted”, “primary”, “groupable”, … and a description can be mentioned. Figure 3 shows two screenshots of the Schema Viewer tool.



```
DATASET_PARAM
id: INT
datasetFK: INT
name: VARCHAR
type: VARCHAR
intValue: INT
floatValue: DECIMAL
stringValue: VARCHAR
timestampValue: TIMESTAMP
booleanValue: BIT
blobValue: BLOB
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

PROJECT
id: INT
name: VARCHAR
description: VARCHAR
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

DATASET
id: INT
projectFK: INT
typeFK: INT
name: VARCHAR
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

DATASET_FILE_BRIDGE
id: INT
datasetFK: INT
fileFK: INT
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

DATASET_TYPE
id: INT
projectFK: INT
name: VARCHAR
description: VARCHAR
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

FILE_TYPE
id: INT
projectFK: INT
name: VARCHAR
description: VARCHAR
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

FILE
id: INT
typeFK: INT
name: VARCHAR
created: TIMESTAMP
createdBy: VARCHAR
modified: TIMESTAMP
modifiedBy: VARCHAR

FILE_VIEW
id: INT
projectName: VARCHAR
datasetName: VARCHAR
fileName: VARCHAR
```

test › DATASET › name

Metadata

Rank: 1003
Description: Logical Dataset Name

☐ Hidden ☐ Admin only ☐ Crypted
☐ Primary ☐ Readable
☐ Statable ☐ Groupable

☑ User free field ☐ Automatic field ☐ « Created » field ☐ « Created by » field
☐ « Modified » field ☐ « Modified by » field

Web link script

```
@NULL
```

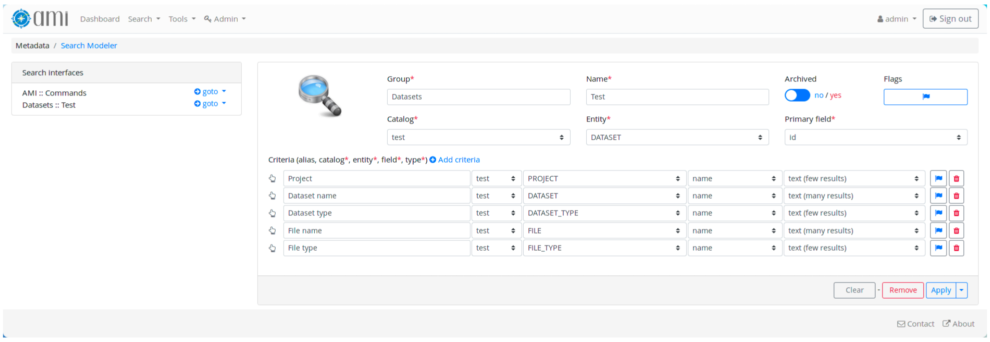
Embedded media

☐ Displayable ☐ Base 64
Media type: NONE
Control: NONE

Fig 3. Screenshot of the AMI Schema Viewer. Visualisation of a database schema (at left) and all the additional metadata associated to the field `test`.`DATASET`.`name` (at right).

3.4 The Search Modeler

The AMI Search Modeler is a tool for designing rich, and across catalogs, search interfaces, see Figure 4. For each interface definition, a set of searchable fields is specified so that the end user is able to select data by criterion on these fields, see Figure 5. For each selection, a MQL query (without any join) is generated and the AMI backend automatically generates a more complex SQL query (with joins).



Group*: Datasets
Name*: Test
Archived: no / yes
Flags
Catalog*: test
Entity*: DATASET
Primary field*: id

Criteria (alias, catalog*, entity*, field*, type*) Add criteria

Project	test	PROJECT	name	text (few results)
Dataset name	test	DATASET	name	text (many results)
Dataset type	test	DATASET_TYPE	name	text (few results)
File name	test	FILE	name	text (many results)
File type	test	FILE_TYPE	name	text (few results)

Fig 4. Screenshot of the AMI Search Modeler. Example of interface definition for searching “Datasets” (see associated database schema Figure 3) by “Project”, “Dataset name”, “Dataset type”, “File name” and “File type”.

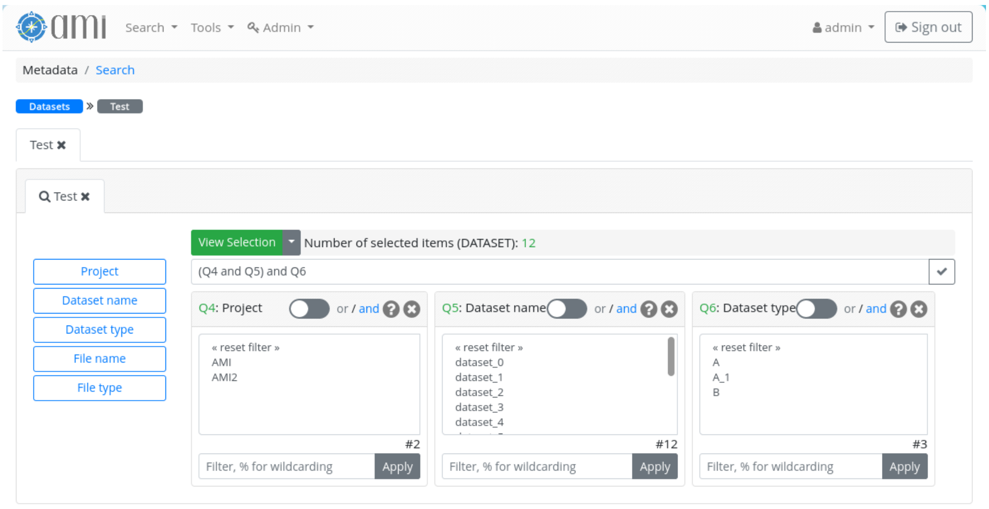


Fig 5. Screenshot of the resulting search interface (see associated database schema Figure 3) for searching "Datasets" by "Project", "Dataset name", "Dataset type", "File name" and "File type". If is selected "Project"='AMI' and "Dataset name"='dataset_2', the resulting MQL query will be: `SELECT * WHERE [`test`.`PROJECT`.`name` = 'AMI'] and (`test`.`DATASET`.`name` = 'dataset_2')`

5 Conclusion

AMI is a well-established and mature metadata ecosystem, proposing services and Web interfaces to more than 2000 active users. The second version of the ecosystem, released in 2018, is very easy to deploy and doesn't require any specialist skills.

This paper showed how to install AMI from the last binary packages. A set of HTML 5 interfaces makes totally graphical the administration of AMI (administration dashboard) and both the Schema Viewer and the Search Modeler tools permit creating rich interfaces for searching data by advanced metadata criteria.

The AMI Team also provides a ready-to-use Docker [10] image. It is planned to publish it on the Docker Hub platform. Additionally, it is seriously considered to deploy the AMI ecosystem with Ansible [11], the new application-deployment tool provided by Red Hat.

The ecosystem is used by the ATLAS at LHC, SuperNEMO [12], Rosetta-Philae [13][14] and NIKA2 [15] experiments.

Acknowledgements

Over the years, we have been helped and supported by many people at CC-IN2P3 and more recently by the Rosetta/Philae collaboration, in particular: Osman Aidel, Philippe Cheynet, Benoît Delaunay, Pierre-Etienne Macchi, Mattieu Puel, Yves Rogez, Mélodie Roudaud and Jean-René Rouet.

References

1. ATLAS Collaboration, *The ATLAS Experiment at the CERN Large Hadron Collider*, *JINST* **3** S08003 doi:10.1088/1748-0221/3/08/S08003 [2008]
2. J. Fulachier, O. Aidel, S. Albrand, F. Lambert, *Looking back on 10 years of the ATLAS Metadata Interface*, proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP) *J. Phys.: Conf. Ser.* **513** 042019 doi:10.1088/1742-6596/513/4/042019 [2013]
3. J. Odier, O. Aidel, S. Albrand, J. Fulachier, F. Lambert, *Evolution of the architecture of the ATLAS Metadata Interface (AMI)*, proceedings of the 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP) *J. Phys.: Conf. Ser.* ***664*** *042040* [2015]
4. J. Odier, F. Lambert, J. Fulachier, *The ATLAS Metadata Interface (AMI) 2.0 metadata ecosystem: new design principles and features*, proceedings of the 23rd International Conference on Computing in High Energy and Nuclear Physics (CHEP) *EPJ Web of Conf.: Conf. Ser.* **214** 05046 doi:10.1051/epjconf/201921405046 [2019]
5. J. Odier, F. Lambert, J. Fulachier, *This Design principles of the Metadata Querying Language (MQL) implemented in the ATLAS Metadata Interface (AMI) ecosystem*, proceedings of the 24th International Conference on Computing in High Energy and Nuclear Physics (CHEP)
6. "JDBC" [software]: https://www.oracle.com/database/technologies/appdev/jdbc.html [accessed 2020-01-10]
7. Open SDK [software]: https://jdk.java.net/ [accessed 2020-01-10]
8. Apache Tomcat [software]: https://tomcat.apache.org/ [accessed 2020-01-10]
9. Java Servlet [software]: https://www.oracle.com/technetwork/java/index-jsp-135475.html [accessed 2020-01-10]
10. Docker [software]: https://www.docker.com/ [accessed 2019-01-10]
11. Ansible [software]: https://www.ansible.com/ [accessed 2019-01-10]
12. SuperNEMO [experiment]: https://supernemo.org/ [accessed 2019-01-10]
13. Rosetta-Philae [experiment]: http://rosetta.esa.int/ [accessed 2019-01-10]
14. Rosetta-Philae [experiment]: https://rosetta.jpl.nasa.gov/ [accessed 2020-01-10]
15. NIKA2 [experiment]: http://ipag.osug.fr/nika2/ [accessed 2020-01-10]