Content

1. The Metadata Catalogue Service (MDC Service)

1.1. Abstract

Advances in computational, storage and network technologies as well as middle ware such as the Globus Toolkit allows scientists to expand the sophistication and scope of data-intensive applications. These applications produce and analyze terabytes and petabytes of data that are distributed in millions of files or objects. To manage these large data sets efficiently, metadata or descriptive information about the data needs to be managed.

1.2. Introduction

The Metadata Catalogue Service (MDC) provides a mechanism for storing and accessing descriptive metadata and allows users to query for data items based on desired attributes. Metadata services are required to support these data intensive applications. Metadata is information that describes data. Metadata services allow scientists to record information about the creation, transformation, meaning and quality of data items and to query for data items based on these descriptive attributes. High performance Grid services are required to support registration and query of metadata information. Metadata Services play a key role in the publication and the discovery and access of data sets.

1.3. Publication

Publication is the process by which data sets and their associated attributes are stored and made accessible to a user community. Many data intensive scientific applications publish data sets as a community. For example, when results of a scientific experiment are obtained, they are calibrated, put into a standard format, and made available or published to the community. Subsequent to the publication, some members of the community may use the Metadata Service to annotate the data set with their own observations using user attributes and make these annotations available to a controlled subset of the community. Scientists may also perform analysis of published data sets to produce new data sets, which are also published along with associated metadata attributes.

1.4. Discovery and Access

Discovery is the process of identifying data items of interest to the user. The Metadata Service allows users to discover data sets based on the value of descriptive attributes, rather than requiring them to know about specific names or physical locations of data items. Typically, the Metadata Service forms one component of data discovery and access in a Grid. Figure 1 illustrates a simple scenario for attribute-based data discovery and access using our Metadata Service, called the MDC [7], the File Catalogue [8], and a Storage Element [9]. In this scenario, a client application first queries the Metadata Service to find data sets with particular attribute values (1). The Metadata Service responds with a list of logical file names for data items with matching attributes (2). Next, the client queries the File Catalogue (3), which returns a list of physical locations for the data content identified by the logical names (4). Finally, the client selects replicas for access, contacts the storage systems where the data items reside (5), and the desired data sets are returned (6).

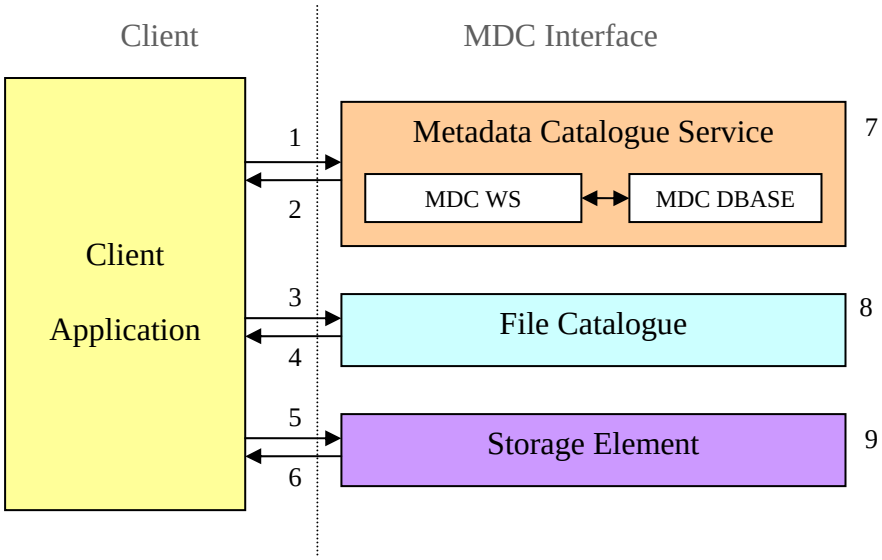


Figure 1: A usage scenario for Metadata Services.

1.5. MDC Service Implementation

The Metadata Catalogue Service uses a web service model. Its implementation uses an Apache web service front end and a MySQL relational database backend. The components of MDC are shown in Figure 2. A client application program issues MDC queries using the MDC API. The MDC client sends these queries to the MDC server using SOAP. The MDC server then interacts with the MySQL database to perform the query, package the results and return them to the MDC client. Finally, the results are returned to the client application.

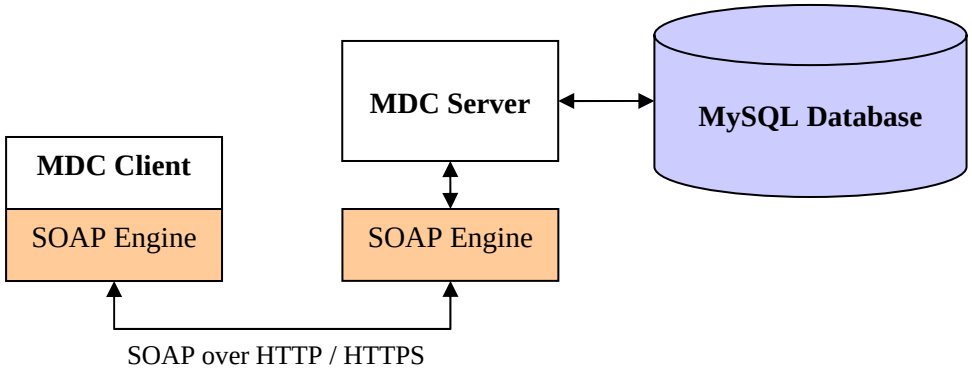


Figure 2: Overview of the Implementation.

.

2. The MDC Core Service

2.1. Overview

The MDC Core provides storing and accessing metadata in XML format and allows users to make queries formulated in XPath expression language for data items based on desired attributes. For a storing/accessing XML data the MDC Core uses a *Hibernate Object-Relation Mapping Tool* (ORM Tool). The idea of ORM consists in the declarative description of conformity between classes and their attributes on the part of objects and tables and columns on the part of a relation database. On the basis of such a description (also called mapping) named mapping, ORM-tools can keep objects in a relation database, load them, and also execute queries formulated in terms of objects, automatically translating them in queries of a database.

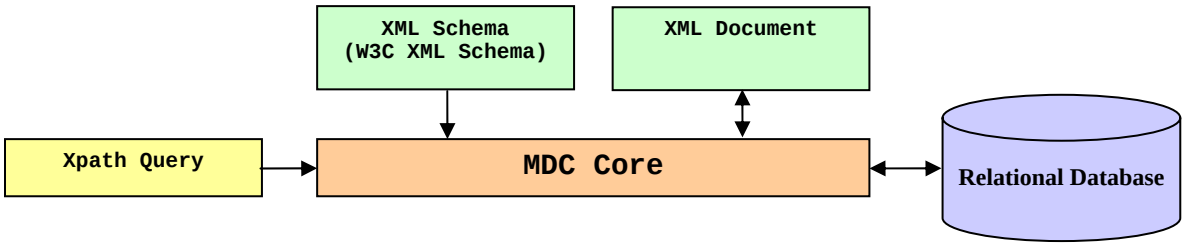


Figure 3: Overview of the MDC Core.

To map between relational tables and columns and Java objects, Hibernate requires an explicit definition of mapping, usually in a form of a XML file. Having this mapping, Hibernate can save objects in a relational database, load them back, query for them etc. To link JAXB objects with Hibernate, we need to produce an object-relational mapping for these objects.
The XDoclet technology allows specifying the mapping properties directly in the source code of the object classes. In this case, source code is annotated with special tags called *"xdoclets"*. XDoclet parses source code and generates the required artifacts (like Hibernate mapping). XDoclet allows developers to remain source-centric instead of supporting several related but separated artifacts (source code and mapping definitions) in parallel. XDoclet is a perfect point to bring JAXB and Hibernate together. JAXB generates source code of Java objects based on the XML Schema. The XDoclet generates object/relational mapping for Hibernate based on the annotated source code.

2.2. The HyperJAXB project

Primary goal of the HyperJAXB project is to link JAXB objects with a relational persistence layer. HyperJAXB provides several extensions for Sun's JAXB RI which adds Hibernate xdoclets to generated sources. Annotated sources of JAXB objects are processed into object/relational mappings for Hibernate. Hibernate also provides tools to produce a database schema for the target database out of the generated mappings. To map between relational tables and columns and Java objects, Hibernate requires an explicit definition of the mapping, usually in a form of XML file. Having this mapping, Hibernate can save objects in a relational database, load them back, query for them etc.

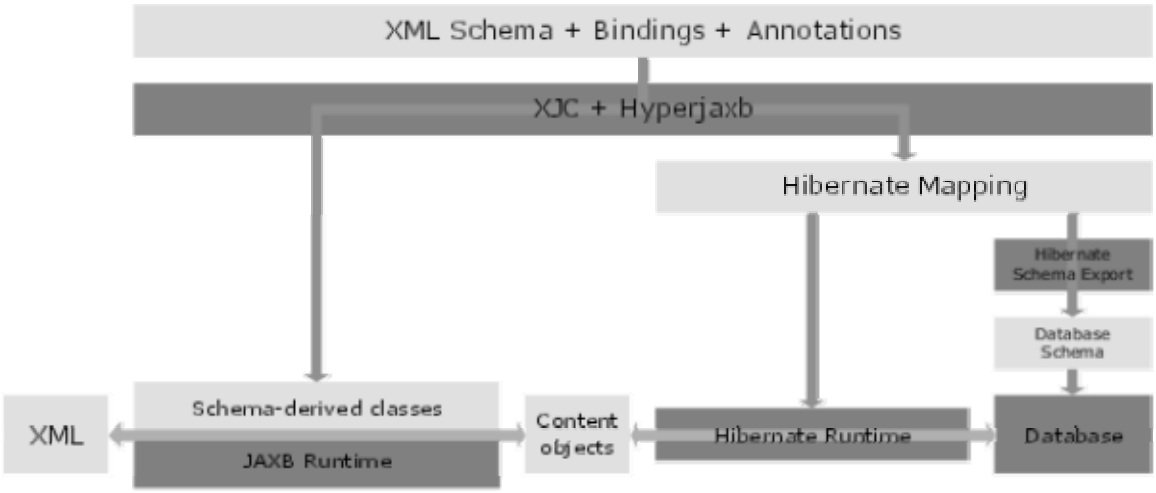


Figure 4: The overall usage of HyperJAXB

The use of HyperJAXB in a project (Figure 4) may be sketched as follows:

Build phase:

- Use binding compiler (XJC) together with HyperJAXB add-on to generate the source code of schema-derived classes and Hibernate mapping.
- Compile the generated classes.
- Use Hibernate toolset to export the database schema.

Runtime phase:

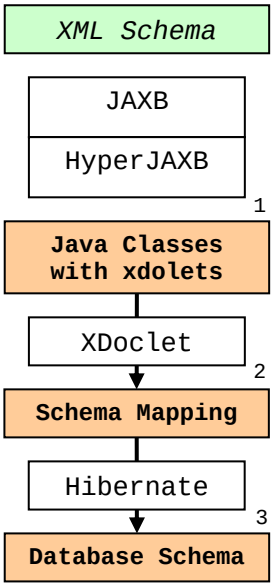
- Use JAXB runtime and schema-derived classes to convert between XML and content objects.
- Use Hibernate to save content objects into the database and load them back.
- Use HQL to formulate queries and process them with Hibernate. Again, use JAXB to marshall results as XML.

2.2.1.Binding XML Schema

Combining JAXB RI, HyperJAXB and Hibernate tools allows you to automatically generate the following artifacts from an XML Schema:

1. Source code of JAXB objects with Hibernate XDoclet annotations;
2. Object/relational mapping for Hibernate;
3. Database schema for the target database.

This XML Schema binding is illustrated in Figure 5.

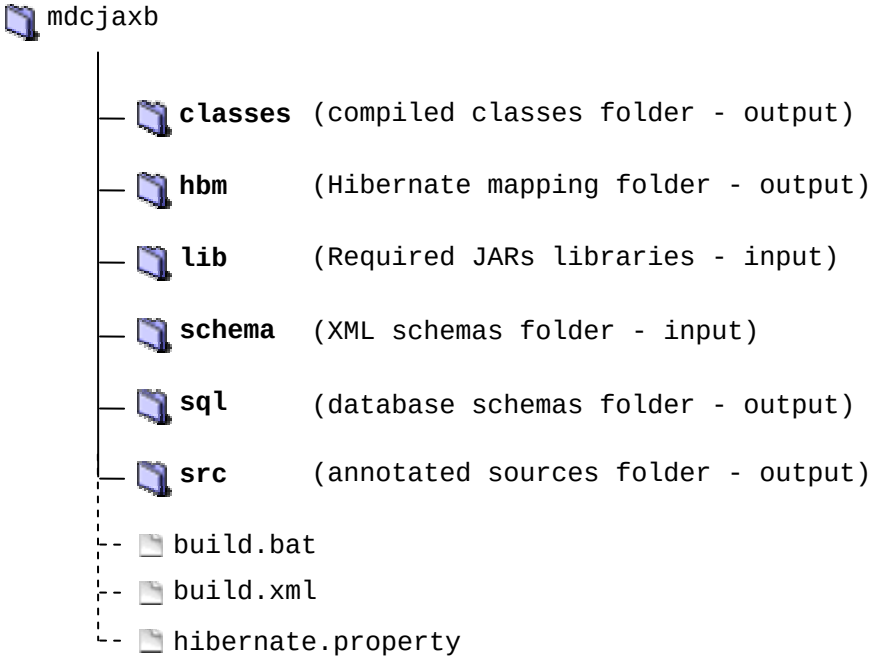


- **JAXB** generates source code of Java objects based on the XML Schema.
- **HyperJAXB** generates xdoclets in the JAXB objects.
- **XDoclet** generates object/relational mapping for Hibernate based on the annotated source code.
- **Hibernate** generates relational database schema.

Figure 5: Binding XML Schema.

2.2.2. Configuration and compilation

The mdcjaxb project has the following structure:



To use mdcjaxb, you will have to update your hibernate.properties to specify the JDBC connectivity.

Configuring Hibernate Properties

hibernate.dialect=net.sf.hibernate.dialect.MySQLDialect
hibernate.connection.driver_class=com.mysql.jdbc.Driver
hibernate.connection.url=jdbc:mysql://localhost/hibernate
hibernate.connection.username=mdavid
hibernate.connection.password=

Next, you have to put the XML Schema(s) into the “schema” directory and just execute the build.bat command file. As a result you will get the following:

- ✓ JAXB classes Annotated with XDoclets in *“src”* directory.
- ✓ Compiled JAXB classes in *“classes”* directory.
- ✓ JAR library in *“lib/mdc”* directory.
- ✓ Hibernate mapping files in *“hbm”* directory.
- ✓ Database schema in *“sql”* directory.

2.2.3. Compilation details

A. Generating annotated JAXB classes

A sample build file fragment is presented below:

```
<target name="generate.sources">
  <taskdef name="xjc"
    classname="com.sun.tools.xjc.XJCTask"
    classpathref="xjc.lib.path"/>
  <mkdir dir="${generated.sources}"/>
  <xjc target="${generated.sources}">
    <arg line="-nv"/>
    <arg line="-extension"/>
    <arg line="-Xhibernate-xdoclets"/>
    <binding dir="${basedir}">
      <include name="binding/*.xml"/>
    </binding>
    <schema dir="${basedir}">
      <include name="schema/*.xsd"/>
    </schema>
  </xjc>
</target>
```

After this step the ***generated.sources*** folder should contain the sources annotated with Hibernate xdoclets.

B. Building generated classes

Building generated classes is not much different from building normal Java classes.

```
<target name="compile" depends="generate.sources">
  <mkdir dir="${classes}"/>
  <javac destdir="${classes}" debug="true"
    srcdir="${generated.sources}"
    classpathref="compile.lib.path">
  </javac>
  <copy todir="${classes}">
    <fileset dir="${generated.sources}">
      <exclude name="**/*.java"/>
    </fileset>
  </copy>
</target>
<target name="jar" depends="compile">
  <mkdir dir="${test.lib.dir}"/>
  <jar jarfile="${test.lib.dir}/${jar.name}.jar" basedir="${classes}"/>
</target>
```

After this step the ***classes*** folder should contain compiled Java classes.

C. Building JAR library

After the Java classes are generated they may be used to create a JAR library.

```
<target name="mdc.build.jar">
 <jar jarfile="${hyperjaxb2.lib.dir}/mdc/ildg-1.2.0.jar"
basedir="${jaxbcommons.generated.classes.dir}"/>
</target>
```

After this step the ***lib/mdc*** folder should contain the JAR library.

D. Generating the Hibernate mapping

Hibernate mapping is generated using the Hibernate XDoclet module. Here is the sample ant target:

```
<target name="generate.hibernate.mapping" depends="jar">
 <taskdef name="hibernatedoclet"
   classname="xdoclet.modules.hibernate.HibernateDocletTask"
   classpathref="hibernatedoclet.lib.path"/>
 <tstamp> <format property="TODAY" pattern="dd-MM-yy"/> </tstamp>
 <mkdir dir="${hibernate.mapping}"/>

  <hibernatedoclet
   destdir="${hibernate.mapping}"
   mergedir="${generated.sources}"
   excludedtags="@version,@author,@todo,@see,@throws"
   addedtags="@xdoclet-generated at ${TODAY},@copyright The XDoclet Team,@author
   XDoclet"
   force="false"
   verbose="false">

   <fileset dir="${generated.sources}">
    <include name="**/*.java"/>
    <exclude name="**/runtime/*.java"/>
   </fileset>

  <hibernate version="2.0"/>
 </hibernatedoclet>
</target>
```

After this step you should find a number of Hibernate mapping files (XXX.hbm.xml) in the ***hibernate*** directory.

E. Customising the Hibernate mapping

Once the Hibernate mapping has been generated, a customizing process step is needed to guarantee compatibility between words reserved by a SQL database and names of columns.

```
<path id="mdc.path">
  <fileset id="mdc.fileset" dir="${hyperjaxb2.lib.dir}/mdc">
   <include name="base-1.0.50.jar" />
   <include name="mdc-full-1.0.80.jar" />
 </fileset>
</path>
<target name="mdc.customize.mapping">
 <java classname="de.desy.md.mdc.util.CustomizeMapping"
 classpathref="mdc.lib.classpath" fork="yes" failonerror="true">
  <arg value="${hyperjaxb2.hbm.dir}" />
   <classpath refid="mdc.lib.classpath" />
 </java>
</target>
```

After this step the Hibernate mapping files (*XXX.hbm.xml*) in the ***hibernate*** directory are updated.

F. Generating the database schema

After the Hibernate mapping is generated it may be used to produce a database schema for the target database.

```
<target name="export.database.schema" depends="generate.hibernate.mapping">
 <taskdef name="schemaexport"
  classname="net.sf.hibernate.tool.hbm2ddl.SchemaExportTask"
  classpathref="schemaexport.lib.path"/>
 <mkdir dir="${database}"/>
  <schemaexport
   properties="${hibernate.properties}"
   quiet="yes"
   text="no"
   drop="no"
   delimiter=";"
   output="${database}/schema.sql">
   <fileset dir="${hibernate.mapping}">
    <include name="**/*.hbm.xml"/>
   </fileset>
  </schemaexport>
</target>
```

This target requires Hibernate properties defined in the *hibernate.properties* file and *XXX.hbm.xml* mapping files in the *hibernate* directory as input. Output is the database schema ***database/schema.sql***.

3. The MDC Web-Service

3.1. Overview

Basically, sharing of objects and XML is reduced to the following tree tasks:

1. Validation of XML Document against XML Schema.
2. Unmarshalling of an XML document into a tree of interrelated instances.
3. Marshalling of content trees into XML documents.

Validation is the process of verifying that an XML document is an instance of a specified XML *schema*. An XML schema defines the content model (also called a *grammar* or *vocabulary*) that its instance documents will represent.

Unmarshalling an XML document means creating a tree of content objects that represents the content and organization of the document. The content tree is not a DOM-based tree. In fact, content trees produced through JAXB can be more efficient in terms of memory use than DOM-based trees.

Marshalling is the opposite of unmarshalling. It creates an XML document from a content tree.

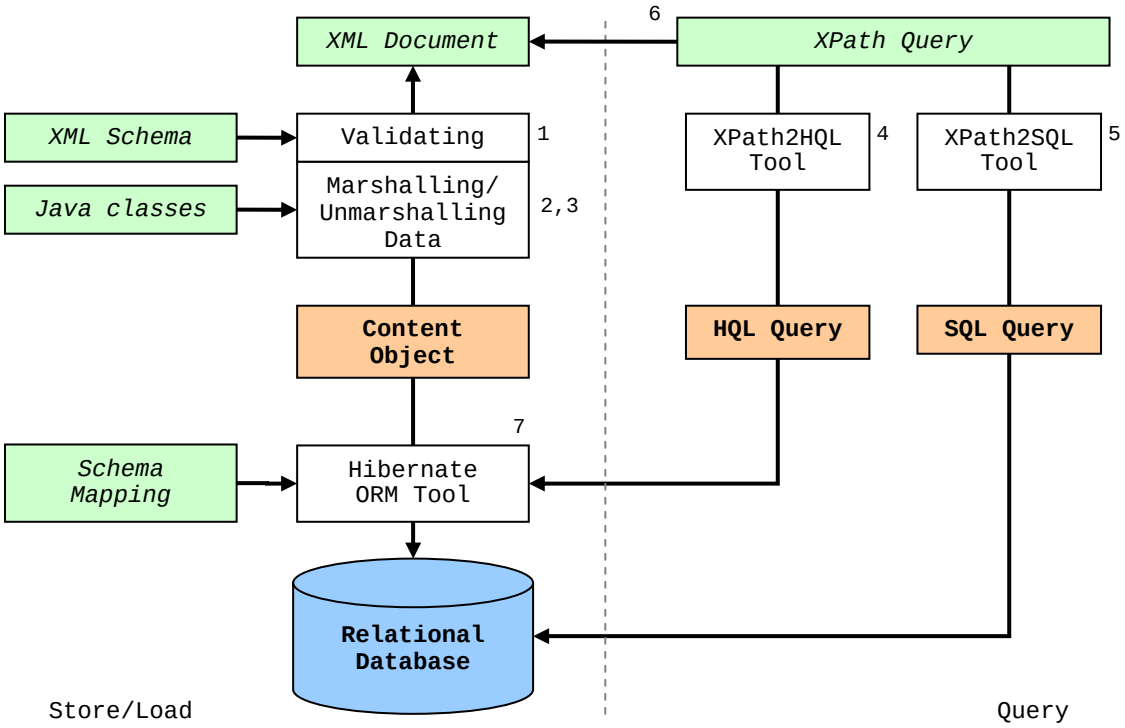


Figure 6: MDC Web-Service.

MDC currently supports XPath as its query language. In some special cases the XPath expression can be automatically translated directly into SQL [5]. In other cases the XPath expression can be translated into HQL [4]. HQL (Hibernate Query Language) is fully object-oriented, understanding notions like inheritance, polymorphism and association. Hibernate is equipped with a powerful query language that looks very much like SQL. Finally, in all other cases the XPath expression has to be applied directly to the marshaled XML Documents [6].

3.2. Security for MDC Web-Service

The security of MDC WS is based on **gLite TrustManager** and contains the following (Figure 7):

1. point-to-point security using SSL based on Grid certificates.
2. authentication using Grid certificates.
3. authorisation using the DN a user's certificate (i.e. map DN to role inside the application).

The **gLite TrustManager** is based on the **EDG TrustManager**. It is a replacement for the SSL implementations which are supplied with web containers and application servers.

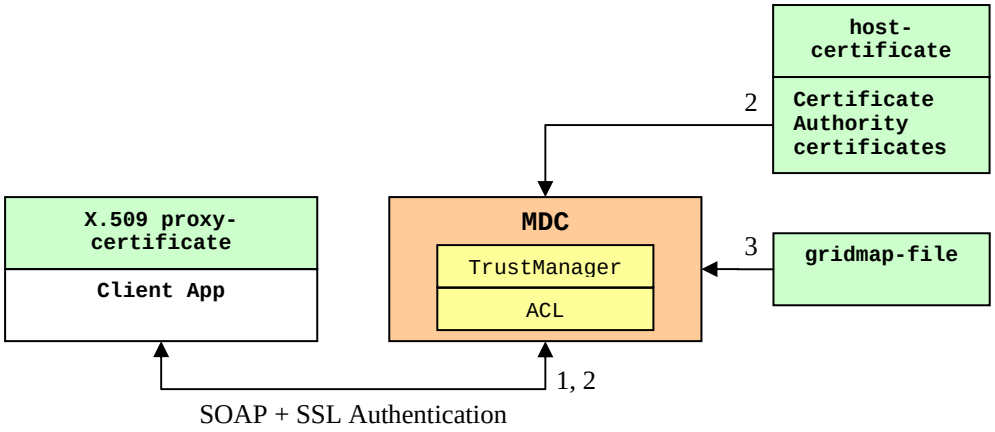


Figure 7: Overview of the MDC Security.

3.3. MDC configuration files

The MDC Core uses the following configuration files:

- ✓ c3p0.properties – c3p0 properties file
- ✓ hibernate.properties – hibernate properties file
- ✓ log4j.properties – log4j properties file
- ✓ settings.xml – mdc properties file
- ✓ server.xml – Tomcat configuration file

c3p0.properties

c3p0 is an easy-to-use library for augmenting traditional (DriverManager based) JDBC drivers with JNDI-bindable DataSources, including DataSources that implement Connection and Statement Pooling, as described by the jdbc3 spec and jdbc2 standard extension.

```
c3p0.acquireRetryAttempts=20
c3p0.acquireRetryDelay=1000
c3p0.breakAfterAcquireFailure=false
c3p0.testConnectionOnCheckin=true
c3p0.automaticTestTable=AutoTestTable
```

hibernate.properties

Hibernate is an object-relational mapping (ORM) solution for the Java language. It provides an easy to use framework for mapping an object-oriented domain model to a traditional relational database.

```
### Query Language ###
hibernate.query.substitutions true 1, false 0, yes 'Y', no 'N'

### MySQL Platform ###
hibernate.dialect org.hibernate.dialect.MySQLMyISAMDialect
hibernate.connection.driver_class com.mysql.jdbc.Driver
hibernate.connection.url jdbc:mysql://mysqlsrv.ifh.de:3306/test_ildg-mdc-dev_1-3-0
hibernate.connection.username mdavid
hibernate.connection.password
hibernate.order_updates true
hibernate.connection.use_compression true
hibernate.connection.cache_result_set_metadata true
hibernate.connection.dont_track_open_resources true
hibernate.connection.dynamic_calendars true
hibernate.connection.zero_date_time_behavior convertToNull
hibernate.connection.rollback_on_pooled_close true
hibernate.query.factory_class org.hibernate.hql.classic.ClassicQueryTranslatorFactory

### C3P0 Connection Pool###
hibernate.c3p0.max_size 150
hibernate.c3p0.min_size 3
hibernate.c3p0.timeout 7200
hibernate.c3p0.max_statements 300
hibernate.c3p0.acquire_increment 3
hibernate.c3p0.idle_test_period 1800

### Plugin ConnectionProvider ###
hibernate.connection.provider_class org.hibernate.connection.C3P0ConnectionProvider

### Miscellaneous Settings ###
## print all generated SQL to the console
hibernate.show_sql false
hibernate.use_sql_comments true
hibernate.cache.region_prefix hibernate.test
```

log4j.properties

log4j is a fast and flexible framework for logging application debugging messages. With log4j it is possible to enable logging at runtime without modifying the application binary. The log4j package is designed so that these statements can remain in shipped code without incurring a heavy performance cost. Logging behaviour can be controlled by editing a configuration file, without touching the application binary. Set the resource string variable to the value of the *log4j.configuration* system property. The preferred way to specify the default initialization file is through the *log4j.configuration* system property. In case the system property *log4j.configuration* is not defined, then set the string variable resource to its default value *"log4j.properties"*.

```
log4j.rootLogger=error, file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.File=${catalina.base}/logs/ws-mdc_1.0.81.log
log4j.appender.file.MaxFileSize=16MB
log4j.appender.file.MaxBackupIndex=16
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{ISO8601} %5p %m%n
log4j.logger.org.glite.security=WARN
log4j.logger.org.apache=WARN
log4j.logger.org.globus=WARN
log4j.logger.org.hibernate=WARN
log4j.logger.com.mchange=WARN
log4j.logger.de.desy.md=DEBUG
#log4j.logger.de.desy.md.mdc.util=INFO
#log4j.logger.de.desy.md.base=INFO
```

settings.xml

The file *settings.xml* contains the global options. The way to specify the default initialization file is through the *mdc.configuration* system property. In case the system property *mdc.configuration* is not defined, then set the string variable resource to its default value *"settings.xml"*.

property	Current value	optional
acl.connection.driver	com.mysql.jdbc.Driver	
acl.connection.url	jdbc:mysql://mysqlsrv.ifh.de:3306/test_ildg-acl-dev	
acl.connection.username	mdavid	X
acl.connection.pswd	password	X
acl.gridmap.file	gridmap-file	
ensemble.class.name	org.lqcd.ildg.qcdml.ensemble1.impl.MarkovChainImpl	
ensemble.xsd.filename	QCDmlEnsemble1.3.0-ldg.xsd	
ensemble.dump.filename	ensemble.dmp	X
config.class.name	org.lqcd.ildg.qcdml.config1.impl.GaugeConfigurationImpl	
config.xsd.filename	QCDmlConfig1.3.0-ldg.xsd	
config.dump.filename	config.dmp	X
xpath.ensemble.uri	/markovChain/markovChainURI/text()	
xpath.config.lfn	/gaugeConfiguration/markovStep/dataLFN/text()	
xpath.config.uri	/gaugeConfiguration/markovStep/markovChainURI/text()	
hibernate.mapping.folder	hbm	
hibernate.properties	hibernate.properties	
mdc.authentication.enabled	true	X
mdc.validation.level	error	X
mdc.input.location	input	X
mdc.output.location	output	X
mdc.read.enabled	true	X
mdc.write.enabled	true	X

server.xml

The file server.xml contains all directives to configure the behaviour of the Tomcat 4 servlet/JSP container. The SSL connector configuration is shown below:

```
<Connector port="8443"
               maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
               enableLookups="false" disableUploadTimeout="true"
               acceptCount="100" debug="0" scheme="https" secure="true"
               sSLImplementation="org.glite.security.trustmanager.tomcat.TMSSLImplementation"
               sslCAFiles=/etc/grid-security/certificates/*.0"
               crlFiles="/etc/grid-security/certificates/*.r0"
               sslCertFile="/etc/certs/globe-meta.ifh.de.hostcert.pem"
               sslKey="/etc/certs/globe-meta.ifh.de.hostkey.pem"
               clientAuth="true" sslProtocol="TLS" />
```

4. Administration of the MDC

The MDC Administration Tool, referred to as list of services, allows you to configure MDC Core remotely.

doSetConfig (name,value) - Privileged access
doGetConfig() - Non-privileged access
doDump() - Privileged access
doRestore() - Privileged access

4.1. doSetConfig(name, value) service

Sets a specified configuration a specified value. The accepted parameters are:

Configuration	values	Default value
mdc.read.enabled	{true, false}	true
mdc.write.enabled	{true, false}	true

Changes made to MDC Core using doSetConfig are not persistent and lost after the restart of the Tomcat Server.

4.2. doGetConfig() service

Gets a list of all configuration settings with specified values.

4.3. doDump() service

Create a dump containing all XML documents stored in the MDC.

The output filenames are defined by the *ensemble.dump.filename* and *config.dump.filename* properties.

Recipe to use this service:

1. Call *doGetConfig* service and check the *mdc.write.enabled* property and if it is equal to *false* then go to point 3.
2. Call *doSetConfig* service with *(mdc.write.enabled, false)* arguments and go to point 1.
3. Call doDump service
4. From time to time call *doGetConfig* as long as *StatusDumpRestore* is not equal to MDC_SUCCESS.
5. Call *doSetConfig* service with *(mdc.write.enabled, true)* arguments.

4.4. doRestore() service

Restore XML documents from previous dump files.

The output filenames are defined by *ensemble.restore.filename* and *config.restore.filename* properties.

Recipe to use this service:

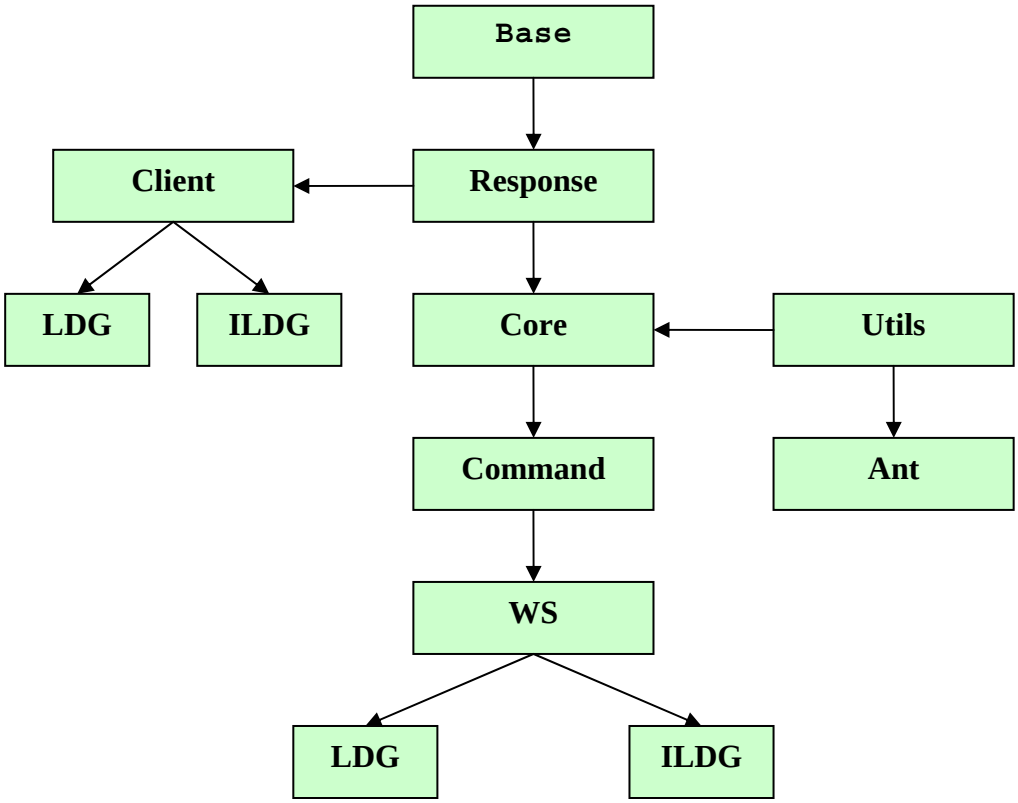
1. Call *doGetConfig* service and check the *mdc.write.enabled/mdc.read.enabled* properties. If they are equal to *false* then go to point 3.
2. Call *doSetConfig(mdc.write.enabled, false)* and *doSetConfig(mdc.read.enabled, false).* Go to point 1.
3. Call doRestore service
4. From time to time call *doGetConfig* as long as *StatusDumpRestore* is not equal to MDC_SUCCESS.
5. Call *doSetConfig(mdc.write.enabled, true)* and *doSetConfig(mdc.read.enabled, true).*

4.5. Migration to a new schema

Recipe to use this service:

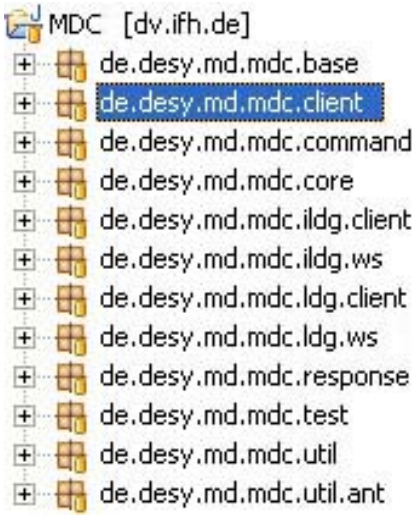
1. Create a dump containing all XML documents.
2. Put new XML Schema(s) into “schema” directory of HyperJAXB2 package and execute the build.bat file.
3. Configure generated hibernate.properties file. Make sure that you use a new **empty** database.
4. The generated database schema can be used to create a new database.
5. Stop running instance of the MDC
6. Update the following files:
   a) JAR library generated by HyperJAXB
   b) Hibernate mapping files and hibernate.properties file
   c) Dump files. If it is necessary adapt them to an updated XML Schema(s).
7. Set mdc.read.enabled and mdc.write.enabled to false
8. Start updated MDC Web-Service
9. Restore XML documents from previous dump files.
10. Set *mdc.read.enabled* and mdc.write.enabled properties via service and also in *settings.xml* file.

5. The structure of MDC source code



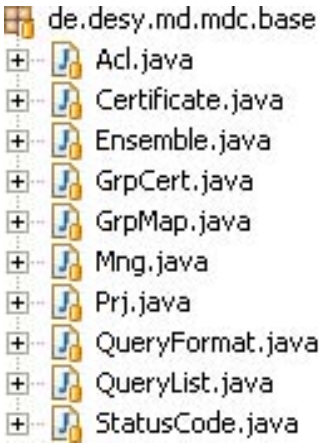
5.1. Java Packages

The MDC Project contains the following Java packages:

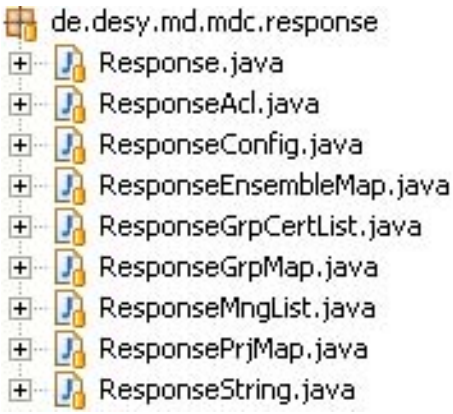


5.2. Java Classes

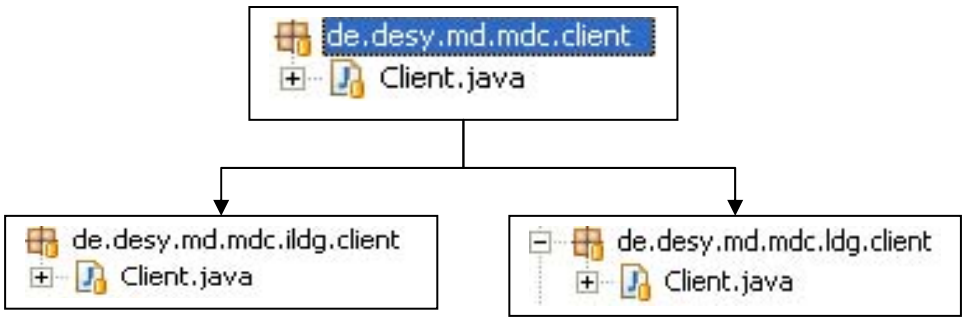
a. The “de.desy.md.mdc.base” package contains all necessary basic classes, which are used to produce a response object. Response objects are Strings that contain XML (SOAP Message). These Messages contain the values returned by Web service method call.



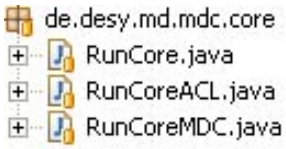
b. The “de.desy.md.mdc.response” package contains all response classes. These classes define the type of the returned values.



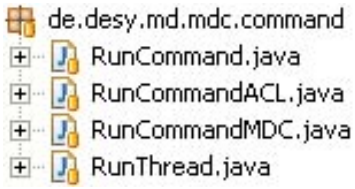
c. The “de.desy.md.mdc.client”, “de.desy.md.mdc.ldg.client” and “de.desy.md.mdc.ildg.client” packages contain the MDC Client classes.



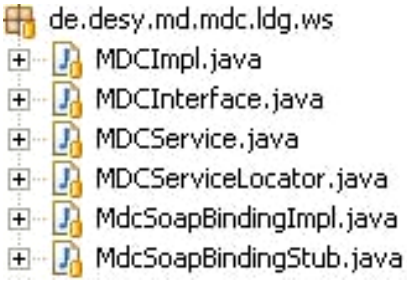
d. The “de.desy.md.mdc.core” package contains the MDC Core classes.

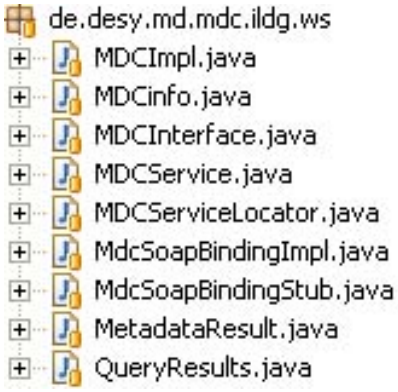


e. The “de.desy.md.mdc.command” package contains the implementation of the WS operations.

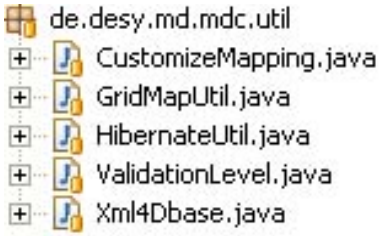


f. The “de.desy.md.mdc.ldg.ws” and “de.desy.md.mdc.ildg.ws” packages contain the MDC WS Interface classes.





g. The “de.desy.md.mdc.util” package contains the following classes.



5.3. List of all used libraries

Filename	Source	Version
activation.jar	Apache Tomcat	5.0–5.5
asm.jar	Hibernate	3.1
axis-ant.jar	Apache Axis	1.3
axis-schema.jar	->>-	->>-
axis.jar	->>-	->>-
base-1.0.60.jar	MDC	1.0.60
bcprov-jdk15-132.jar	TrustManager	1.9
c3p0-0.9.0.jar	C3P0	0.9.0
cglib-2.1.3.jar	Hibernate	3.1
cog-jglobus.jar	COG Kit	1.2
commons-beanutils.jar	Apache Commons Bean Utils	1.7.x
commons-collections.jar	Apache Commons Collections	2.0
commons-discovery-0.2.jar	Apache Commons Discovery	0.2
commons-lang.jar	Apache Commons Lang	2.1
commons-logging-1.0.4.jar	Apache Commons Logging	1.0.4
dom4j.jar	dom4j	1.5.x
glite-security-trustmanager.jar	TrustManager	1.9
glite-security-util-java.jar	->>-	->>-
hibernate3.jar	Hibernate	3.1
hyperjaxb2.jar	HyperJAXB	2.0
ildg-1.3.0.jar	Annotated JAXB library	1.3.0
jax-qname.jar	HyperJAXB	2.0
jaxb-api.jar	JWSDP	2.0
jaxb-impl.jar	->>-	->>-
jaxb-libs.jar	->>-	->>-
jaxb-xjc.jar	->>-	->>-
jaxp-api.jar	->>-	->>-
jaxrpc.jar	->>-	->>-
jsr173_1.0_api.jar	dom4j	1.5.x
jta.jar	Apache Tomcat	5.0–5.5
log4j-1.2.13.jar	Log4Java	1.2.x
mdc-full-1.0.81.5.jar	MDC	1.0.81
mysql-connector-java-3.1.12-bin.jar	MySQL Connector	3.1.12
relaxngDatatype.jar	JWSDP	2.0
resolver.jar	->>-	->>-
saaj.jar	Apache Axis	1.3
wsdl4j.jar	->>-	->>-
xsdlib.jar	HyperJAXB	2.0

5.4. Glossary

XML Schema published as a **W3C** Recommendation, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C. XML Schema can be used to express a schema: a set of rules to which an XML document must conform in order to be considered 'valid' according to that schema.

XML Path Language (XPath) is terse (non-XML) syntax for addressing portions of an XML document. This allows consumers of XML to query it and directly access any part of it. XPath can address a single word or character within a document or can refer to whole sections at once. An XPath expression can be applied to an XML document, node or node-set; it may return null, a string, a number, an XML node or an XML node-set.

Hibernate is an object-relational mapping (ORM) solution for the Java language. It is free, open source software that is distributed under the GNU Lesser General Public License. It provides an easy to use framework for mapping an object-oriented domain model to a traditional relational database.

Hibernate Query Language (HQL) is a query language used by Hibernate enabled applications. HQL is similar to SQL with the difference that it is case sensitive (except for the java class names and property names that it uses). The main use of HQL is to enable Hibernate framework generate sql queries and execute it against underlying database. HQL is different from SQL also in the fact that it uses Classes (which maps to tables in the databases) and properties (which correspond to the name of the columns in the table) instead of tables and columns. HQL also supports Polymorphism, Associations, and is much less verbose than SQL.

Java Architecture for XML Binding (JAXB) allows Java developers to create and edit XML using familiar Java objects. JAXB is particularly useful when the specification is complex and changing. In such a case, regularly changing the XML Schema definitions to keep them synchronised with the Java definitions can be time consuming and error prone.

XDoclet is an open-source code generation library which enables Attribute-Oriented Programming for Java via insertion of special Javadoc tags.

Ant is the premier build-management tool for Java environments. Ant is part of Jakarta, the Apache Software Foundation's open source Java project repository. Ant is written entirely in Java, and is platform independent. Using XML, a Java developer describes the modules involved in a build, and the dependencies between those modules. Ant then does the rest, compiling components as necessary in order to build the application.

Web service is a software system designed to support interoperable machine-to-machine interaction over a network. This definition encompasses many different systems, but in common usage the term refers to those services that use SOAP-formatted XML envelopes and have their interfaces described by WSDL.

gLite TrustManager is a pure Java-based solution for validation of GSI-style (X.509) certificate paths used in SSL/TLS connections to secure Web Services.

5.5. Acknowledgements

- A MDC Service for Data Intensive Applications at http://www.globus.org/alliance/publications/papers/mcs_sc2003.pdf
- JAXB and XJC; Java Architecture for XML Binding (JAXB) provides a convenient way to bind an XML schema to a representation in Java code. You can find Jaxb at http://java.sun.com/xml/jaxb/
- Hibernate; Hibernate is a high performance object/relational persistence and query service for Java. You can find Hibernate at http://www.hibernate.org
- JXPath - provides utilities for manipulating Java classes that conform to the JavaBeans naming conventions using the XPath syntax. The library also supports maps, DOM and other object models. Jakarta Commons can be found at http://jakarta.apache.org
- HyperJAXB; HyperJAXB is and add-on for Sun's reference implementation of JAXB (Java Architecture for XML Binding). HyperJAXB automatically generates Hibernate mappings for classes produced by JAXB effectively providing JAXB objects with relational persistence capabilities. HyperJAXB can be found at https://hyperjaxb.dev.java.net/
- Security for Web Services at http://www.physics.gla.ac.uk/metadata/index.php/SecuringWebServices
- Apache Tomcat http://tomcat.apache.org/ and Apache Ant at http://ant.apache.org/
- XDoclet Attribute Oriented Programming at http://xdoclet.sourceforge.net/xdoclet/index.html