ELSAG DATAMAT
A Finmeccanica Company

P-50638/DSASGT-1762-10/00

25 June, 2010

CONTRACT N. ESA/ESRIN 22508/09/I-LG

OPGW Software Installation Document
HMA FOLLOW ON TASK 4 – ORDER

SUMMARY

This document specifies the procedures for building, configuring and deploying the Ordering & Programming Gateway – OPGW system on the runtime environment.

Prepared by: Daniele MARCHIONNI

Certifies that the current document was prepared analyzing input documents and standards applicable to the project; cooperated with reference roles (OBS) reporting to a technical responsible.

(Project Manager)

Verified by: Daniele MARCHIONNI

Certifies: consistency and completeness of the content with respect to contract and company/divisional procedures and other outputs generated by previous phases; the correspondence to applicable standards; absence of orthographic and typographical errors.

(Project Manager)

Technical Approval: Daniele MARCHIONNI

Certifies validity of technical decisions and document coherence with both upper-level documentation and documents correlated to the expected project phase.

(Project Manager)

Quality Approval: Raffaele BARBATI

Certifies that: document complies with defined requirements; the process which generated it was performed as foreseen; quality records exist for the quality activities performed; any action generated was closed.

(SPA Manager)

Authorized by: Mario FONTI

Certifies that the document complies with directives and constraints set by Purchaser and that any possible commitment arisen by document is compatible with project constraints; authorizes usage and distribution.

(SW Engineering Manager)

Distribution

Pier Giorgio MARCHETTI – ESA
Yves COENE – Spacebel
Uwe VOGES – con terra

Release and Edition Register

Edition	Release	Date	Amendments description	Author
1	0	25/06/2010	First Issue	D. MARCHIONNI

Table of Contents

Indexes of Figures

Indexes of Tables

1 Introduction

This document specifies the procedures for building, configuring and deploying the Ordering & Programming Gateway system on the runtime environment.

At this stage of the project, HMA FO MTR, this document has to be considered as an early draft of the final OPGW Software Installation Document that will be delivered with the OPGW software: in fact the development phase is foreseen in the next phase of the HMA Follow-on project and then no precise installation instructions can be provided now.

However, OPGW is not a new system, but it is a new version of the software already implemented in previous projects:

- HMA-I
- HMA-E

and then the procedures for building, installing, configuring and operating the software will not be very different from those already specified in the currently available OPGW installation document [RD-07].

Stated that, in the next chapters the information for installing the software is reported highlighting the sections / topics that are subject to major changes in the final version of the document, together with obviously an overall revision of the document itself and of all detailed procedures, instructions, and parameters.

The intended readership is:

- system engineering staff supervising and taking part in the development of the HMAFO project;
- ESA technical staff.

Document content:

- the chapter 1 lists the information that can be found in this document;
- the chapter 2 provides the list of applicable and reference documents;
- the chapter 3 provides the terms and abbreviations used in this document;
- the chapter 4 provides the hardware and software resources needed both for building and running the system;
- the chapter 5 provides the procedure for building the software;
- the chapter 6 specifies the procedure for configuring and operating the software.

2 Applicable and Reference documents

The following table provide the list of applicable documents:

Id.	Title	Reference	Issue	Date
[AD-01]	EARTHNET ONLINE XML FRONT-END INTERFACE CONTROL DOCUMENT	EOLI-XML-006-ICD	2.8	21 Jan 2008
[AD-02]	OGC™ Catalogue Services Specification 2.0 Extension Package for ebRIM (ISO/TS 15000-3) Application Profile: Earth Observation Products	OGC 06-131r6	0.2.4	07 May 2008
[AD-03]	Application schema for Earth Observation products	0GC 06-080r4	0.9.3	21 Jul 2008
[AD-04]	ECSS – Space engineering – Software	ECSS-E-ST-40C		06 Mar 2009
[AD-05]	Ordering Services for Earth Observation Products	OGC 06-141r2	0.9.5	02 Jul 2010
[AD-06]	EARTHNET ONLINE XML FRONT-END: ORDER AND ON-LINE ACCESS EXTENSION INTERFACE CONTROL DOCUMENT	EOLI/Order-XML-ICD	3.4	07 Apr 2008
[AD-07]	OPGW Software Requirements Specification Document	P50638/DSASGT-0082-10/00	1.1	25 Jun 2010
[AD-08]	Proposal for HMA Follow On Task 4 – Order	EF000D135/DSASGT-0501-09	1.0	13 Mar 2009
[AD-09]	Multi Mission User Information Services MMOHS/UMS XML ICD, UMS Import/Export XML ICD	UMS-MMOHS-XML-ICD	1.2	31 May 2006
[AD-10]	M2AS MMOHS IMPORT/EXPORT XML ICD	OSME-USMP-SEDA-IS-08-2059	1.1	21 Nov 2008
[AD-11]	User Management Interfaces for Earth Observation Services	OGC 07-118r4	0.0.6	29 Jan 2010
[AD-12]	SOFTWARE DEVELOPMENT PLAN FOR HMA FOLLOW ON TASK 4 – ORDER	P50638/DSASGT-2995-09/00	1.0	20 Nov 2009
[AD-13]	SOFTWARE PRODUCT ASSURANCE PLAN FOR HMA FOLLOW ON TASK 4 – ORDER	P-P50638/DSAQUD-3046-09/00	1.0	20 Nov 2009

Table 2-1. Applicable Documents

The following table provide the list of reference documents:

Id.	Title	Reference	Issue	Date
[RD-01]	HSQLDB Web Site	http://www.hsqldb.org/		
[RD-02]	EOLI-SA Configuration File Interface Control Document	VEGA.EOLI-SA.ICD.042	1.17	16 Apr 2008
[RD-03]	OPGW Software Requirements Specification Document	OSME-USMP-SEDA-RS-08-1855	1.2	20 Mar 2009
[RD-04]	Apache XML Security	http://santuario.apache.org/Java/index.html		
[RD-05]	OPGW SDD	P50638/DSASGT-0083-10/00	1.1	25 Jun 2010
[RD-06]	Prototype Operations Concept	HMA-PR-SPB-EN-0001	1.0	12 Aug 2006
[RD-07]	OPGW Software Transfer Document & Operator / Administrator Manual	OSME-SWEN-SEDA-PR-09-0048	1.4	04 Dec 2009

Table 2-2: Reference Documents

3 Terms, definition and abbreviations terms

Acronym	Definition
API	Application Programmer's Interface
ASCII	American Standard for Code Information Interchange
ASN.1	Abstract Syntax Notation One
BNF	Backus-Naur Form
CDR	Critical Design Review
CM	Contributing Mission
COTS	Commercial off-the-shelf
DAIL	Data Access Integration Layer
DAP	Data Access Portfolio
DB	Database
EO-A	Enhanced On-line Access
EOLI	Earthnet On-line Interactive
EoliSA	Earthnet On-line Interactive and Stand-Alone Client: main user interface to the catalogue and ordering on-line services
ESA	European Space Agency
GMES	Global Monitoring for Environment and Security
GSC	GMES Space Component
GSDR	Ground Segment Design Review
GSOV	Ground Segment Operation Validation.
HMA	Heterogeneous Mission Accessibility
ICD	Interface Control Document
M2EOS	Multi Mission Earth Observation Services
M2AS	Multi Mission authorization Service
M2BS	Multi Mission Browse Server
M2CS	Multi Mission Catalogue Server
MMOHS	Multi-Mission Order Handling System
MTA	Medium Term Archive
MUIS	Multi-Mission User information Services
N/A	Not Applicable
NRT	Near Real Time
OMT	Object Modelling Technique
OPGW	ESA G/S Ordering and Programming Gate-Way
OR	Operational qualification Review
OSME	Operational Support, Maintenance and Evolution Contract
PDR	Preliminary Design Review
PDS	Payload Data Segment
SEDA	Serco / Elsag Datamat Consortium
SOAP	Simple Object Access Protocol
SWG	Standard Working Group
TBC	To Be Confirmed
TBD	To Be Defined
TBW	To Be Written
TCP	Transmission Control Protocol
UML	Unified Modelling Language
UMS	User Management System
USMP	User Services & Mission Planning

Table 3-1: Acronyms and definitions

4 Software Delivery and Prerequisites

4.1 Delivery

The software package is delivered as a couple of Zip files named:

- OPGW.X.Y.zip
  Which contains the OPGW developed software.
- OPGW _COTS.X.Y.zip
  This contains the COTS used by OPGW:
  - Java Development Kit 1.6.x
  - Ant 1.x
  - Tomcat 6.x
  - HSQLDB 1.8.0.7
  - XMLBEANS 2.x

The precise versions of all the used COTS will be defined in the next stage of the project.

4.2 Building & Running Environment

Hardware resources:

- INTEL Centrino 1.83 Ghz or higher;
- 1 GB RAM;
- 200 MB of free disk space to hold the OPGW software and the development and runtime tools;

Software resources:

- Supported Operating Systems:
  - RedHat Linux ES 5 Update 2.
  - Windows XP.

5 Software Build Procedure

5.1 Unpacking the Delivery Package

Place the delivery kit packages into the $HOME directory of the chosen user of the building host.

Unpack the packages, both the delivered software and the COTS packages using the appropriate commands:

```
UNIX>unzip OPGW.X.Y.zip
UNIX>unzip OPGW_COTS.X.Y.zip
```

For Windows XP:

Choose the installation directory <OPGW_INST_DIR> and unpack the software using the preferred archiving tool (e.g. winzip, Winrar, etc.)

5.2 JDK 1.6.x installation

The precise Java 1.6.x version to be sued will be defined in the next phase of the project.

Install the JDK extracted by the COTS package by issuing the following commands:

```
UNIX>chmod a+x jdk-<VERSION>-linux-i586.bin
UNIX>./jdk-<VERSION>-linux-i586.bin
```

Agree the license agreement and extract the JDK software.

To complete the installation several environment variables defined in

```
$HOME/OPGW/conf/env.ksh
```

have to be set:

- Define the JAVA_HOME with the path where JDK has been installed:
  ```
  UNIX>export JAVA_HOME=$HOME/jdk<VERSION>
  ```
- Put JAVA_HOME/bin in $PATH variable. In order to avoid to use other java installations already present in the system, update the PATH variable in this way:
  ```
  UNIX>export PATH=$JAVA_HOME/bin:$PATH
  ```

To activate above definitions, run the following command from the shell prompt:

```
UNIX>. $HOME/OPGW/conf/env.ksh
```

To check whether Java and related environment variables are correctly set, run the following command:

```
UNIX>java -version
```

which shall return the downloaded version (e.g. 1.5.0_17).

Windows XP:

- install Java running the file:
  - jdk-<VERSION>-windows-i586-p.exe
- to set JAVA_HOME env variable with the path where JDK has been installed:
  - open Control Panel; go to Advanced tab; click on "Environment variables"; in "System Variables" set: JAVA_HOME=C:\Programmi\Java\jdk<VERSION>

To check whether Java and related environment variables are correctly set, run the following command:

```
UNIX>java -version
```

which shall return the downloaded version (e.g. 1.5.0_17).

5.3 Tomcat 6.x installation

The TOMCAT 6.x precise version to be used will be defined in the next stage of the project. In the following the instruction for installing Tomcat V6.0.18 are reported.

Unpack TOMCAT software issuing the following command:

```
UNIX>unzip apache-tomcat-6.0.18.zip
```

To complete the installation several environment variables defined in

```
$HOME/OPGW/conf/env.ksh
```

have to be set:

- Set the CATALINA_HOME variable for pointing to the desired Apache Tomcat installation:

```
UNIX>export CATALINA_HOME=$HOME/apache-tomcat-6.0.18
```

- Add $CATALINA_HOME/bin in $PATH.
- Set the TOMCAT_HOME with value referring to the Tomcat installation directory:

```
UNIX>export TOMCAT_HOME=$HOME/apache-tomcat-6.0.18
```

To activate above definitions, run the following command from the shell prompt:

```
UNIX>. $HOME/OPGW/conf/env.ksh
```

Make sure whether the executables have the correct execution flag set:

```
UNIX>cd $CATALINA_HOME/bin
UNIX>chmod a+x *.sh
```

To check whether Tomcat has been correctly installed, try to start-up and shutdown the server:

```
UNIX>$CATALINA_HOME/bin/startup.sh
UNIX>$CATALINA_HOME/bin/shutdown.sh
```

Windows XP:

Run the windows service installer:

- apache-tomcat-6.0.18.exe
- follow the instruction of the installation wizard, leave the **default port 8080**, leave **admin/admin for the administration account**
- Set the needed environment variables (Control Panel; go to Advanced tab; click on "Environment variables"; in "System Variables"):
  - CATALINA_HOME=<APACHE TOMCAT INST DIR>
  - TOMCAT_HOME=<APACHE TOMCAT INST DIR>

5.4 Ant 1.x installation

The Ant 1.x precise version to be used will be defined in the next stage of the project. In the following the instruction for installing Ant V1.6.5 are reported.

Unpack ANT software issuing the following command:

```
UNIX>unzip apache-ant-1.6.5-bin.zip
```

To complete the installation several environment variables defined in

$HOME/OPGW/conf/env.ksh

have to be set:

```
UNIX>export ANT_HOME=$HOME/apache-ant-1.6.5
```

And add Ant's bin subdirectory to the executable PATH list:

```
UNIX>export PATH=$PATH:$ANT_HOME/bin
```

To activate above definitions, run the following command from the shell prompt:

```
UNIX>. $HOME/OPGW/conf/env.ksh
```

Make sure whether the executables have the correct execution flag set:

```
UNIX>cd $ANT_HOME/bin
UNIX>chmod a+x ant antRun
```

To check whether Tomcat has been correctly installed, try to start the tool

```
UNIX>ant -version
```

Windows XP:

- Choose the installation directory <ANT_INST_DIR> and unpack the software using the preferred archiving tool (e.g. winzip, Winrar, etc.)
- Set the needed environment variables (Control Panel; go to Advanced tab; click on "Environment variables"; in "System Variables"):
  - ANT_HOME=<ANT_INST_DIR>
  - Add $ANT_HOME/bin to the PATH env var.

5.5 RDBMS Installation

The RDBMS used for the OPGW is the HSQLDB: lightweight RDBMS written in Java, which suits well the needs of this software.

Unpack HSQLDB software issuing the following command:

```
UNIX>unzip hsqldb.zip
```

To finalise the installation the following steps have to be followed:

- Customize the $HOME/OPGW/conf/env.ksh defining the following environment variables:
  - Set the environment variable HSQLDB_HOME to the parent directory of the lib directory that contains hsqldb.jar:

    ```
    unix>export HSQLDB_HOME=$HOME/hsqldb
    ```

  - Set and export the environmental variable CLASSPATH to the value of HSQLDB_HOME (as described above) plus "/lib/hsqldb.jar":

    ```
    UNIX>export CLASSPATH=$HSQLDB_HOME/lib/hsqldb.jar:$CLASSPATH
    ```

  - Add $HSQLDB_HOME/bin to $PATH:

    ```
    UNIX>export PATH=$HSQLDB_HOME/bin:$PATH
    ```

- activate above definitions, by run the following command from the shell prompt:

  ```
  UNIX>. $HOME/OPGW/conf/env.ksh
  ```

- Make sure whether the executables have the correct execution flag set:

  ```
  UNIX>cd $HSQLDB_HOME/bin
  UNIX>chmod a+x *.sh
  ```

- Start-up the RDBMS:

  ```
  UNIX>cd $HSQLDB_HOME/DatabaseFiles
  ```

```
UNIX>nohup java org.hsqldb.Server &
```
or
```
UNIX>nohup ../bin/StartDb.sh &
```

Windows XP:

- Set the needed environment variables (Control Panel; go to Advanced tab; click on "Environment variables"; in "System Variables"):
  - HSQLDB_HOME=<OPGW_INST_DIR>\hsqldb
  - Add `HSQLDB_HOME\lib\hsqldb.jar` to the CLASSPATH env var
  - Add `HSQLDB_HOME\bin` to the PATH env var

```
Dos>..\bin\StartDb.bat
```

WARNING: the Database must be started in the specified directory. In fact HSQLDB creates / re-use the database files in the directory where it is started. Then activating the HSQLDB from different directories you will have different databases!

- Set-up the database schema:
  - Activate the SQL shell:
    ```
    UNIX> SqlTool.sh
    or
    dos>SqlTool.bat
    ```
  - Run the schema creation script:
    ```
    SQL>\i $HOME/OPGW/db_schema/order_database.sql
    SQL>\i $HOME/OPGW/db_schema/users_database.sql
    ```
  - For checking whether the DB tables have been actually created:
    ```
    Sql>\dt
    ```
  - For checking the structure of created DB TABLES:
    ```
    Sql>\d <DB TABLE NAME>
    ```

    The names of the DB tables to be checked will be provided in the next phase of the project.

  - To exit from the SQL prompt issue the following command:
    ```
    Sql>\q
    ```

For shut down the RDBMS the following command has to be issued:

```
UNIX>ShutdownDb.sh
```
Or
```
dos> ShutdownDb.bat
```

5.6 Environment setup

In order to have automatic set of previously defined environment variables, the file:

```
$HOME/OPGW/conf/env.ksh
```

has to be activated at user log in. It can be accomplished by calling it from the:

```
$HOME/.bashrc
```

5.7 Build Configuration

5.7.1 Tomcat Configuration

The TOMCAT 6.x precise version to be used will be defined in the next stage of the project. In the following the instruction for configuring Tomcat V6.0.18 are reported.

On a dedicated host, the default configuration is often appropriate, however on a shared host, this is rarely the case.

The file:

$CATALINA_HOME/conf/server.xml

contains the configuration options to change.

The line that looks like:

```
<Server port="8005" shutdown="SHUTDOWN" debug="0">
```

defines a control port for administration purposes (e.g. to send the shutdown command to the server).

The <Connector> tags define service listening end point (by default the server is listening to the port 8080):

```
<Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000"
redirectPort="8443"/>
<Connector port="8009" protocol="AJP/1.3" redirectPort="8443"/>
```

Hereafter a brief description for a simple HTTPS configuration. For more detailed explanation of TOMCAT configuration please refer to TOMCAT web site.

5.7.1.1 HTTPS and decryption configuration

5.7.1.1.1 Prepare the Certificate Keystore

To create a new keystore from scratch, containing a single self-signed Certificate, execute the following from a terminal command line:

Windows: `%JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA`

Unix: `$JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA`

(The RSA algorithm should be preferred as a secure algorithm, and this also ensures general compatibility with other servers and components.)

This command will create a new file, in the home directory of the user under which you run it, named

.keystore

To specify a different location or filename, add the -keystore parameter, followed by the complete pathname to your keystore file, to the keytool command shown above. You will also need to reflect this new location in the server.xml configuration file, as described later. For example:

```
Windows: %JAVA_HOME%\bin\keytool -genkey -alias tomcat -keyalg RSA \
            -keystore \path\to\my\keystore
Unix: $JAVA_HOME/bin/keytool -genkey -alias tomcat -keyalg RSA \
            -keystore /path/to/my/keystore
```

After executing this command, you will first be prompted for the keystore password. The default password used by Tomcat is "changeit" (all lower case), although you can specify a custom password if you like. You will also need to specify the custom password in the server.xml configuration file, as described later.

Next, you will be prompted for general information about this Certificate, such as company, contact name, and so on. This information will be displayed to users who attempt to access a secure page in your application, so make sure that the information provided here matches what they will expect.

Finally, you will be prompted for the key password, which is the password specifically for this Certificate (as opposed to any other Certificates stored in the same keystore file). You MUST use the same password here as was used for the keystore password itself. (Currently, the keytool prompt will tell you that pressing the ENTER key does this for you automatically.)

5.7.1.1.2 Prepare key pair for SAML Token decryption

In order to allow OPGW to decrypt and verify digital signatures of incoming SAML Token please perform the following steps:

- Create a keypair (RSA type) and store them in a JKS keystore using keytool (http://java.sun.com/javase/6/docs/technotes/tools/solaris/keytool.html)

  unix>keytool -genkey -alias <alias> -keyalg RSA -keystore <path>

  where:

  <alias> is the key alias used in "key_alias" in hma.properties config file (see §6.1.2)

  <path> is the path used in "keystore_path" in hma.properties config file (see §6.1.2)

  The information requested during the key creation should be the same of the # SAML Encryption properties of "hma.properties" config file (see §6.1.2).

  (Note: the same command is used in a windows based architecture.)

- Export the public key just generated to OPGW clients (e.g. DAIL) using keytool (http://java.sun.com/javase/6/docs/technotes/tools/solaris/keytool.html)
- Configure "hma.properties" file "SAML Encryption properties" with the parameter of the keypair and the keystore just generated.
- Copy "data" directory and its subdirectories (provided in OPGW software delivery) in Apache Tomcat home directory (e.g. for windows "C:\Program Files\Apache Software Foundation\Tomcat 6.0") and, for Linux installation make sure that it is reachable by Apache Tomcat process by giving the correct rights to the related user.
- Restart OPGW.

5.7.1.1.3 Edit the Tomcat Configuration File

There are several ways for configuring TOMCAT, depending on whether the APR (Apache Portable Runtime) is used or not or whether the SSL is being used or not. Anyway, to avoid auto configuration, you can define which connector to use by specifying a classname in the protocol attribute.
To define a Java connector, regardless if the APR library is loaded or not do:

<-- Define a blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->

<!--

<Connector protocol="org.apache.coyote.http11.Http11Protocol"

port="8443" minSpareThreads="5" maxSpareThreads="75"

enableLookups="true" disableUploadTimeout="true"

```
        acceptCount="100"  maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="${user.home}/.keystore" keystorePass="changeit"
        clientAuth="false" sslProtocol="TLS"/>
-->
<-- Define a non-blocking Java SSL Coyote HTTP/1.1 Connector on port 8443 -->
<!--
<Connector protocol="org.apache.coyote.http11.Http11NioProtocol"
        port="8443" minSpareThreads="5" maxSpareThreads="75"
        enableLookups="true" disableUploadTimeout="true"
        acceptCount="100"  maxThreads="200"
        scheme="https" secure="true" SSLEnabled="true"
        keystoreFile="${user.home}/.keystore" keystorePass="changeit"
        clientAuth="false" sslProtocol="TLS"/>
-->
```

The first one has been verified during OPGW validation.

Note that, on Windows platform, the above configurations expect to find the .keystore file in C:

For checking whether the configuration is OK, start-up the server and:

- check whether **Catalina<date>.log** file does not report errors
- connect to **https://<hostname>:8443** (default HTTPS port) and check whether the home page is showed.

5.7.2 OPGW Configuration

To build the OPGW software, customize the following script:

$HOME/OPGW/conf/env.ksh

to define the OPGW_HOME environment variable to point to the directory where OPGW sources are installed:

```
unix>export OPGW_HOME=$HOME/
```

to define the OPGW_CURRENT environment variable to point to the directory where src, scripts, conf, etc. directories are put:

```
unix>export OPGW_CURRENT=$HOME/OPGW
```

To activate above definition, run the following command from the shell prompt:

```
UNIX>. $HOME/OPGW/conf/env.ksh
```

Windows XP:

- Set the environment variables (Control Panel; go to Advanced tab; click on "Environment variables"; in "System Variables"):

```
OPGW_HOME=<OPGW_INST_DIR>
```

Check the OPGW software build file:

$OPGW_CURRENT/build.xml

which is the file for building the software, for optionally setting the suitable values for:

- "web.context", which defines the path at which the software gets installed in tomcat webapp directory. Default value is "opgw".
- "http.port", which is the port where the server is listening to. Default value is 8080.

5.8 Building the Software

5.8.1 OPGW Build

Once the configuration above is finalized, to build the prototype software issue the following commands:

```
UNIX>cd $OPGW_HOME
UNIX>ant
```

It will show all available build options:

- release: to make an OPGW release
- install: to install the OPGW application files
- uninstall: to uninstall the OPGW application files
- installServices: to install OPGW services
- uninstallServices: to remove OPGW services

```
UNIX>ant install
```

This command compiles the source files and copy all required files into the destination directories of the TOMCAT installation.

After the software has been built, the service implemented by the gateway has to be registered into the TOMCAT server by issuing the following command (**make sure the server is up and running**):

```
UNIX>ant installServices
```

To verify whether the deployment has been correctly installed, open a standard web browser and connect to the link:

"http://<OPGW IP>:<http.port>/<web.context>/hma/web/Test.jsp"

By default:

"http://<hostname>:8080/opgw/hma/web/Test.jsp"

Or in case HTTPS has been enabled, connect to:

"https://<hostname>:8443/opgw/hma/web/Test.jsp"

Verify whether the Test page is properly displayed. Note that the configuration has not been completed yet and then the system is not fully functional.

Make sure that the scripts in $OPGW_CURRENT/scripts are executable by issuing the command:

```
UNIX>cd $OPGW_CURRENT/scripts
UNIX>chmod a+x *.sh
```

To un-register the service and remove all OPGW software from the TOMCAT server the following commands have to be issued:

```
UNIX>ant uninstallServices
UNIX>ant uninstall
```

6 Configuration and Operation

This section describes the configuration files and the procedures for operating the ESA GMES HMA Prototype.

6.1 Configuration

The $OPGW_CURRENT/conf directory contains several files needed for configuring the system.

To make effective the changes to the configuration files, the gateway software has to be deployed again:

UNIX>cd $OPGW_HOME

UNIX>ant install

UNIX>ant installServices

OPGW services & operations can be called to the following URLs:

HTTP:

- http://<server name>:<server port>/opgw/services/<service name>/<service operation>
- SOAP Action: <service operation>

or HTTPS:

- https://<server name>:<server port>/opgw/services/<service name>/<service operation>
- SOAP Action: <service operation>

Where:

- <server name>: name of the server where OPGW is installed
- <server port>: TCP / IP port where the server is listening to (by default 8080 for HTTP and 8443 for HTTPS)
- <service name>: either **ordering**
- <service operation>: the required service operation (e.g. GetCapabilities, GetOptions, Submit, etc.

Warning:

In case of Window installation, the path separator is "\" and in all configuration variables it must be set in this way "****"

Example:

WRONG:

```
capabilities_path=C:\Program Files\Apache Software Foundation\Tomcat
6.0\webapps\opgw\WEB-INF\classes\capabilities\ordering
```

CORRECT:

```
capabilities_path=C:\\Program Files\\Apache Software Foundation\\Tomcat
6.0\\webapps\\opgw\\WEB-INF\\classes\\capabilities\\ordering
```

6.1.1 conf.properties

- capabilities_path: this value is path to a directory where various "capabilities" version documents of Ordering Service that describe the various supported versions are grouped.

- dsm2hma_getstatus_xsl_path: this value holds the path to an XSL file used to translate GetStatus responses from DSM ones to equivalent HMA ones.

Further parameters will be identified in the next phase of the project.

6.1.2 hma.properties

This file contains both parameters used by the testing framework originally developed as the skeleton testing service and OPGW Specific parameters. The first set of parameters includes paths and filenames which shouldn't be changed. The second set of parameters has to be configured for the correct operation of OPGW:

- DSM_URL: a URL which describes where the M2EOS service is
- EOLI_ORDER_SUBMIT_SOAP_ACTION: Name of the SOAP action to be set for EOLI product order requests
- EOLI_ORDER_MONITOR_SOAP_ACTION: Name of the SOAP action to be set for EOLI order monitor requests
- EOLI_ORDER_SUBMIT_ACQUISITION_SOAP_ACTION: Name of the SOAP action to be set for EOLI acquisition order requests
- EOLI_INVENTORY_SEARCH_SOAP_ACTION: Name of the SOAP action to be set for EOLI search requests
- EOLI_INVENTORY_PRESENT_SOAP_ACTION: Name of the SOAP action to be set for EOLI present requests.
- DEFAULT_ORDERING_WSA_REPLY_TO: this is the default reply to address for Submit request of ordering service. This is mostly for testing purposes.

  Default value:

  http://localhost:8080/opgw/services/ordering
- CONFIG_FILES_ROOT: directory under the root directory of OPGW installation where the config files are put.

The following options are required to connect to the HSQL instance used for the asynchronous response of the Submit request:

- DATABASE_DRIVER: JDBC driver class to use
- DATABASE_CONNECTION_STRING: URL formatted connection string describing host, port and database
- DATABASE_USERNAME: the username to use for the connection
- DATABASE_USERPASSWORD: the password to use for the connection

The following ones specify the polling time-out for CheckOrders tools:

- ORDER_POLLING_TIME_OUT: interval in seconds at which to check the M2EOS for order status updates.

The following parameters specify the path for OrderOption XML file and stylesheet.

- product_mapping_path: this value holds the path to an xml file which contains collection options
- product_mapping_xsl_path: this value holds the path to an XSL file which is used to translate from the previous file to an HMA GetOptions response.

The following parameters specify the configuration for SAML token decryption (see §5.7.1.1.2)

- encryption: if set to "true" SAML token decryption functionalities are enabled
- verify_signature: if set to "true" SAML token digital signature verification functionalities are enabled
- keystore_path: path of the keystore containing the private key used to decrypt incoming SAML Token
- keystore_password: password to enter in the keystore
- key_alias: alias of the private key used by OPGW to decrypt incoming SAML Token
- key_password: password of the private key used by OPGW to decrypt incoming SAML Token

Further parameters will be identified in the next phase of the project.

6.1.3 log4j.properties

This file is used to configure the logging parameters for the service, most of the options should be left as they are, unless particular custom logging options are required, and only the path of the log file should be changed. This option is:

log4j.appender.HMALOG.file=<path to logfile>

the suggested path is the hma/logs subdirectory inside the OPGW webapp directory of the TOMCAT installation:

<TOMCAT_HOME>\webapps\opgw\hma\logs\hma_log.xml

6.2 Operation

The OPGW includes the following components:

- **Apache TOMCAT**: It is the main component of the OPGW: it is the environment hosting the Ordering & Programming services;
- **HSQLDB**: it is the RDBMS that maintains the orders submitted by clients which support asynchronous notification.
- **CheckOrders**: it is a stand-alone application which is in charge of monitoring the status of orders submitted to the ESA GS and send back notifications to clients which are waiting for asynchronous notifications.
- **Order Options Converter Tool**: it is a tool in charge of extracting and converting in HMA format the order options stored in the EOLI SA ServiceDirectory.xml file.
- **ImportUsers:** script for loading MMOHS Users in the OPGW User Database.

To start-up the OPGW the following components have to be started-up:

- Apache TOMCAT
- HSQLDB
- CheckOrders

6.2.1 Apache Tomcat

For activating the ordering & programming services, the TOMCAT server has to be activated. To do this the following command has to be issued:

UNIX>$CATALINA_HOME/bin/startup.sh

DOS>%CATALINA_HOME%\bin\startup.bat

To shutdown the server the following command has to be issued:

UNIX>$CATALINA_HOME/bin/shutdown.sh

```
DOS>%CATALINA_HOME%\bin\shutdown.bat
```

6.2.2 HSQLDB

The HSQLDB instance is started by executing the following command:

```
UNIX>cd $HSQLDB_HOME/DatabaseFiles
UNIX>nohup java org.hsqldb.Server &
```

or

```
UNIX>nohup ../bin/StartDb.sh &
```

To shutdown the RDBMS issue the following command:

```
UNIX>ShutdownDb.sh
```

To connect to the Database to issue SQL commands:

```
UNIX>SqlTool.sh
```

6.2.3 CheckOrders Tool

For checking the status of Orders submitted to ESA GS there are the following scripts:

```
Directory: $OPGW_CURRENT/scripts/

UNIX>CheckOrders.sh
DOS>CheckOrders.bat
```

Before using the scripts, check whether the path for classes and config files are properly set in the scripts themselves:

OPGW_CLASS_ROOT=$OPGW_CURRENT/axis/WEB-INF/lib

OPGW_CLASS_ROOT2=$OPGW_CURRENT/axis/WEB-INF/classes

OPGW_CONFIG_DIR=$OPGW_CURRENT/conf

To activate the tool issue the following commands:

```
UNIX>cd $OPGW_CURRENT/scripts
UNIX>CheckOrders.sh
or
DOS>CheckOrders.bat
```

To shutdown the tool is sufficient to kill the application using the **kill** command.

Note: it is important to notice that this script **must** be launched from the **`$OPGW_CURRENT/scripts/`**

6.2.4 Order Options Converter Tool

This tool is a python script, which requires the following COTS:

- Python 2.4.4 or 2.5 (www.python.org)
- lxml library 1.1.2 (http://codespeak.net/lxml/)

To install these tools follow the documentation provided on the reported web sites.

To activate the Order Option Import Tool, issue the following commands:

```
UNIX>cd $OPGW_CURRENT/scripts
UNIX>python extractProductMapping.py <EOLI SA Servicedirectory.xml file path> >
productMapping.xml
```

Then move the produced productMapping.xml to the path specified in the conf.properties configuration file.

The produced file is composed of a set of XML blocks having the following elements:

- <collectionId>, by default set with "SET_THE_COLLECTION_ID"
- <product>, set with the product extracted from ServiceDirectory.xml
- <sensor>, set with the product extracted from ServiceDirectory.xml
- <productOrderOptionsId>, set with the order option group name of ServiceDirectory.xml
- <option>, specifying the processing options extracted from ServiceDirectory.xml
- <sceneSelectionOption>, specifying the scene selection options extracted from ServiceDirectory.xml
- <productDeliveryOptions>, specifying the delivery options extracted from ServiceDirectory.xml
- <qualityOfService>, specifying the quality of service extracted from ServiceDirectory.xml

The field collection Id shall be manually configured specifying the correct value starting from the <product> and <sensor> element filled with the EOLI SA values.

6.2.5 Logs

The OPGW generates different log files:

- Tomcat default log file:
  - $TOMCAT_HOME/logs/catalina.out
- Ordering & Programming services specific logs:
  - $ TOMCAT_HOME/webapps/opgw/hma/logs/hma_log.xml

    This file logs the activity of the services (what operations are triggered, the invocation of MUIS-DSM, etc.).

    This path can be configured within the log4j.properties file.
  - $ TOMCAT_HOME/webapps/opgw/hma/logs/

    In this directory the received and sent messages are logged.

7 Appendix

7.1 TOMCAT (6.0.18) server.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  Licensed to the Apache Software Foundation (ASF) under one or more
  contributor license agreements.  See the NOTICE file distributed with
  this work for additional information regarding copyright ownership.
  The ASF licenses this file to You under the Apache License, Version 2.0
  (the "License"); you may not use this file except in compliance with
  the License.  You may obtain a copy of the License at

      http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
-->
<!-- Note:  A "Server" is not itself a "Container", so you may not
     define subcomponents such as "Valves" at this level.
     Documentation at /docs/config/server.html
 -->
<Server port="8005" shutdown="SHUTDOWN">
        <!--APR library loader. Documentation at /docs/apr.html -->
        <Listener className="org.apache.catalina.core.AprLifecycleListener" SSLEngine="on"/>
        <!--Initialize Jasper prior to webapps are loaded. Documentation at /docs/jasper-howto.html -->
        <Listener className="org.apache.catalina.core.JasperListener"/>
        <!-- JMX Support for the Tomcat server. Documentation at /docs/non-existent.html -->
        <Listener className="org.apache.catalina.mbeans.ServerLifecycleListener"/>
        <Listener className="org.apache.catalina.mbeans.GlobalResourcesLifecycleListener"/>
        <!-- Global JNDI resources
       Documentation at /docs/jndi-resources-howto.html
  -->
        <GlobalNamingResources>
                <!-- Editable user database that can also be used by
         UserDatabaseRealm to authenticate users
    -->
                <Resource name="UserDatabase" auth="Container" type="org.apache.catalina.UserDatabase" description="User
database that can be updated and saved" factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
pathname="conf/tomcat-users.xml"/>
        </GlobalNamingResources>
        <!-- A "Service" is a collection of one or more "Connectors" that share
       a single "Container" Note:  A "Service" is not itself a "Container",
       so you may not define subcomponents such as "Valves" at this level.
       Documentation at /docs/config/service.html
   -->
        <Service name="Catalina">
                <!--The connectors can use a shared executor, you can define one or more named thread pools-->
                <!--
    <Executor name="tomcatThreadPool" namePrefix="catalina-exec-"
        maxThreads="150" minSpareThreads="4"/>
    -->
                <!-- A "Connector" represents an endpoint by which requests are received
         and responses are returned. Documentation at :
         Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
         Java AJP  Connector: /docs/config/ajp.html
         APR (HTTP/AJP) Connector: /docs/apr.html
         Define a non-SSL HTTP/1.1 Connector on port 8080
    -->
```

```
                <Connector port="8080" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443"/>
                <!-- A "Connector" using the shared thread pool-->
                <!--
    <Connector executor="tomcatThreadPool"
               port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
    -->
                <!-- Define a SSL HTTP/1.1 Connector on port 8443
        This connector uses the JSSE configuration, when using APR, the
        connector should be using the OpenSSL style configuration
        described in the APR documentation -->
<!-- DM: original example for HTTPS configuration
    <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
               maxThreads="150" scheme="https" secure="true"
               clientAuth="false" sslProtocol="TLS"
               keystoreFile="${user.home}/.keystore" keystorePass="changeit" />
-->
<Connector protocol="org.apache.coyote.http11.Http11Protocol"
           port="8443" minSpareThreads="5" maxSpareThreads="75"
           enableLookups="true" disableUploadTimeout="true"
           acceptCount="100"  maxThreads="200"
           scheme="https" secure="true" SSLEnabled="true"
           keystoreFile="${user.home}/.keystore" keystorePass="changeit"
           clientAuth="false" sslProtocol="TLS"/>


                <!-- Define an AJP 1.3 Connector on port 8009 -->
                <Connector port="8009" protocol="AJP/1.3" redirectPort="8443"/>
                <!-- An Engine represents the entry point (within Catalina) that processes
        every request.  The Engine implementation for Tomcat stand alone
        analyzes the HTTP headers included with the request, and passes them
        on to the appropriate Host (virtual host).
        Documentation at /docs/config/engine.html -->
                <!-- You should set jvmRoute to support load-balancing via AJP ie :
    <Engine name="Standalone" defaultHost="localhost" jvmRoute="jvm1">
    -->
                <Engine name="Catalina" defaultHost="localhost">
                        <!--For clustering, please take a look at documentation at:
          /docs/cluster-howto.html  (simple how to)
          /docs/config/cluster.html (reference documentation) -->
                        <!--
      <Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
      -->
                        <!-- The request dumper valve dumps useful debugging information about
          the request and response data received and sent by Tomcat.
          Documentation at: /docs/config/valve.html -->
                        <!--
      <Valve className="org.apache.catalina.valves.RequestDumperValve"/>
      -->
                        <!-- This Realm uses the UserDatabase configured in the global JNDI
          resources under the key "UserDatabase".  Any edits
          that are performed against this UserDatabase are immediately
          available for use by the Realm.  -->
                        <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
resourceName="UserDatabase"/>
                        <!-- Define the default virtual host
          Note: XML Schema validation will not work with Xerces 2.2.
       -->
                        <Host name="localhost" appBase="webapps" unpackWARs="true" autoDeploy="true"
xmlValidation="false" xmlNamespaceAware="false">
                                <!-- SingleSignOn valve, share authentication between web applications
            Documentation at: /docs/config/valve.html -->
                                <!--
        <Valve className="org.apache.catalina.authenticator.SingleSignOn" />
        -->
```

```
                <!-- Access log processes all example.
        Documentation at: /docs/config/valve.html -->
                <!--
    <Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
           prefix="localhost_access_log." suffix=".txt" pattern="common" resolveHosts="false"/>
    -->
                </Host>
            </Engine>
        </Service>
</Server>
```