
# eIDAS-Node Installation, Configuration and Integration

# Quick Start Guide

Version 1.3

## Document history

| Version | Date | Modification reason | Modified by |
|---|---|---|---|
| 1.0 | 26/11/2015 | Modifications to align with the eIDAS technical specifications. | DIGIT |
| 1.1.0 | 29/06/2016 | Modifications due to installation changes related to architectural and stability improvements<br><br>Update of the deployments configuration and related libraries | DIGIT |
| 1.2.0 | 20/01/2017 | Configuration and stability improvements, please see Version 1.2.0 Release Notes. | DIGIT |
| 1.3.0 | 05/05/2017 | Modifications to align with changes in Technical Specifications version 1.1. For details please see the Version 1.3.0 Release Notes. | DIGIT |

## Table of contents



 

## List of abbreviations

The following abbreviations are used within this document.

| Abbreviation | Meaning |
|---|---|
| AT | The ISO 3166 International Standard country code for Austria. |
| DE | The ISO 3166 International Standard country code for Germany. |
| eIDAS | electronic Identification and Signature. The Regulation (EU) N°910/2014 governs electronic identification and trust services for electronic transactions in the internal market to enable secure and seamless electronic interactions between businesses, citizens and public authorities. |
| IdP | Identity Provider. An institution that verifies the citizen's identity and issues an electronic ID. |
| LoA | Level of Assurance (LoA) is a term used to describe the degree of certainty that an individual is who they say they are at the time they present a digital credential. |
| MS | Member State. |
| SAML | Security Assertion Markup Language |
| SP | Service Provider |

## List of definitions

The following definitions are used within this document.

| Term | Meaning |
|---|---|
| eIDAS-Node | An eIDAS-Node is an application component that can assume two different roles depending on the origin of a received request. See eIDAS-Node Connector and eIDAS-Node Proxy Service. |
| eIDAS-Node Connector | The eIDAS-Node assumes this role when it is located in the **Service Provider's** Member State. In a scenario with a Service Provider asking for authentication, the eIDAS-Node Connector receives the authentication request from the Service Provider and forwards it to the eIDAS-Node of the citizen's country. |
| eIDAS-Node Proxy Service | The eIDAS-Node assumes this role when it is located in the **citizen's** Member State. The eIDAS-Node Proxy Service receives authentication requests from an eIDAS-Node of another MS (their eIDAS-Node Connector). The eIDAS-Node Proxy-Service also has an interface with the national eID infrastructure and triggers the identification and authentication for a citizen at an identity and/or attribute provider. |

# 1. Introduction

This document describes how to quickly install a Service Provider, eIDAS-Node Connector, eIDAS-Node Proxy Service and IdP from the distributions in this release package. The distributions provide preconfigured eIDAS-Node modules for running on each of the supported application servers (Glassfish, Tomcat, JBoss, WebLogic and WebSphere).

Details of the setup and configuration of the sample eIDAS-Nodes, are included in the *eIDAS-Node Installation, Configuration and Integration Manual*.

This document is divided into the following sections:

- Section 1 – *Introduction*;
- Section *2* – *Release content* lists the files delivered with this release and describes their contents;
- Section *3* – *Overview of the preconfigured demo eIDAS-Node packages* illustrates the setup of the configurations provided with this distribution;
- Section *4* – *Demo eIDAS-Node set up and configuration* describes step-by-step how to install the demo configuration;
- Section 5 – *Specific configuration* provides information on how the setup can be changed to suit your needs;
- Section *6* – *Compiling the modules from the source* describes how to rebuild the Maven project if necessary;
- Section *7* — *Enabling logging* describes how to enable audit logging of the communications between eIDAS-Node Proxy Service and Connector.

## 2. Release content

For information on the changes in this release, please see the current Release Notes.

The deliverable consists of the following zip files:

| Deliverable | Description |
|---|---|
| EIDAS-1.3.0.zip | Distribution version 1.3.0 of the sample eIDAS-Node |
| EIDAS-Sources-1.3.0.zip | Source files (Maven project) of the sample eIDAS-Node including an example of implementation of a SP (service provider) and IdP (Identity Provider). |
| EIDAS-Binaries-Glassfish-1.3.0.zip | Deployable war files of a preconfigured eIDAS-Node for a Glassfish server (including IdP.war, EidasNode.war, SP.war) |
| EIDAS-Binaries-Jboss-1.3.0.zip | Deployable war files of a preconfigured eIDAS-Node for a JBoss server (including IdP.war, EidasNode.war, SP.war) |
| EIDAS-Binaries-Tomcat-1.3.0.zip | Deployable war files of a preconfigured eIDAS-Node for a Tomcat server (including IdP.war, EidasNode.war, SP.war) |
| EIDAS-Binaries-Was-1.3.0.zip | Deployable war files of a preconfigured eIDAS-Node for a WebSphere server (including IdP.war, EidasNode.war, SP.war) |
| EIDAS-Binaries-Wls-1.3.0.zip | Deployable war files of a preconfigured eIDAS-Node for a WebLogic server (including IdP.war, EidasNode.war, SP.war) |

## 3. Overview of the preconfigured demo eIDAS-Node packages

This distribution provides an example configuration in which each supported server represents one country providing an eID service. For the purpose of this demo, fictitious countries are used (CA, CB, CC, CD, CF).

The following table illustrates the setup of the configurations provided with this distribution.

| Application Server | version | Default host | Default port | Country | Description |
|---|---|---|---|---|---|
| **Tomcat** | 7*, 8 | localhost | 8080 | CA | Country A |
| **Glassfish** | 3*, 4 | localhost | 8081 | CB | Country B |
| **JBoss** | 6*, 7 | localhost | 8085 | CC | Country C |
| **WebLogic** | 10.3.6*, 12.1.2 | localhost | 7001 | CD | Country D |
| **WebSphere/**<br><br>**WebSphere Liberty Profile** | 8.5.5*<br><br>8.5.5.4 | localhost | 9080 | CF | Country F |

* Default build server provided with the binaries

## 4. Demo eIDAS-Node set up and configuration

Each example eIDAS-Node package is preconfigured to use 'localhost' as hostname and a default http listening port; see the table in section 3. The http listening port of your application server must be adapted according to these default values.

If you need to change these default values, refer to section *5.1 — Changing the default hostname or http port* for details.

To set up and configure the demo, perform the following steps:

1. Install Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files which contain no restriction on cryptographic strengths:

2. Download the Java Cryptography Extension (JCE) Unlimited Strength Policy Files from Oracle:

    - For Java 7: http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

    - For Java 8: http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

3. Uncompress and extract the downloaded zip file (it contains README.txt and two jar files).

4. For the installation, please follow the instructions in the README.txt file.

5. If you are using Tomcat it is necessary to:

    - Create a folder named `shared` in `$TOMCAT_HOME`.

    - Create a subfolder named `lib` in `$TOMCAT_HOME\shared`

    - Edit the file `$TOMCAT_HOME\conf\catalina.properties` and change the property `shared.loader` so that it reads:

    ```
    shared.loader=${catalina.home}\shared\lib\*.jar.
    ```

    Note that for Linux OS it should be:

    ```
    shared.loader=${catalina.home}/shared/lib/*.jar)
    ```

6. Copy the files below to the `endorsed` directory on the application server (Tomcat, Glassfish, JBoss). These jars may be found under `AdditionalFiles` directory in the binary for your application server.

    ```
    endorsed\xml-apis-1.4.01.jar
    endorsed\resolver-2.9.1.jar
    endorsed\serializer-2.7.2.jar
    endorsed\xalan-2.7.2.jar
    endorsed\xercesImpl-2.11.0.jar
    ```

7. It is necessary to increase the default JVM memory settings. Set the following JVM parameter in the startup script of your application server `XX:MaxPermSize=512m`.

8. Local directory or directories must be defined in order to store the configuration files and the test keystores. These directories need to be defined either as OS/AS environment variables or command-line parameters:

   `EIDAS_CONFIG_REPOSITORY` for EidasNode,
   `SPECIFIC_CONFIG_REPOSITORY` for Specific,
   `SP_CONFIG_REPOSITORY` for SP
   `IDP_CONFIG_REPOSITORY` for IdP.

   It is also possible to use only one common directory for all the modules.JVM command line example:

```
DEIDAS_CONFIG_REPOSITORY=c:/Pgm/projects/configEidas/glassfish_v13/ -
DSPECIFIC_CONFIG_REPOSITORY=c:/Pgm/projects/configEidas/glassfish_v13/specifi
c/ -DSP_CONFIG_REPOSITORY=c:/Pgm/projects/configEidas/glassfish_v13/sp/ -
DIDP_CONFIG_REPOSITORY=c:/Pgm/projects/configEidas/glassfish_v13/idp/
```

   By default the following configuration file structure must be present:

```
eidas.xml
encryptionConf.xml
EncryptModule_Connector.xml
EncryptModule_Service.xml
hazelcast.xml
saml-engine-additional-attributes.xml
SamlEngine.xml
SamlEngine_Connector.xml
SamlEngine_Service.xml
SignModule_Connector.xml
SignModule_Service.xml
idp/EncryptModule_IdP.xml
idp/idp.properties
idp/IdPSamlEngine.xml
idp/saml-engine-additional-attributes.xml
idp/saml-engine-eidas-attributes.xml
idp/SamlEngine_IdP.xml
idp/SignModule_IdP.xml
idp/user.properties
keystore/demoKeys.jks
keystore/eidasKeyStore.jks
keystore/eidasKeyStore_Connector_CA.jks
keystore/eidasKeyStore_Connector_CB.jks
keystore/eidasKeyStore_Connector_CC.jks
keystore/eidasKeyStore_Connector_CD.jks
keystore/eidasKeyStore_Connector_CF.jks
keystore/eidasKeyStore_IDP_CA.jks
keystore/eidasKeyStore_IDP_CB.jks
keystore/eidasKeyStore_IDP_CC.jks
keystore/eidasKeyStore_IDP_CD.jks
keystore/eidasKeyStore_IDP_CF.jks
keystore/eidasKeyStore_METADATA.jks
keystore/eidasKeyStore_Service_CA.jks
keystore/eidasKeyStore_Service_CB.jks
keystore/eidasKeyStore_Service_CC.jks
keystore/eidasKeyStore_Service_CD.jks
keystore/eidasKeyStore_Service_CF.jks
keystore/eidasKeyStore_SP_CA.jks
keystore/eidasKeyStore_SP_CB.jks
keystore/eidasKeyStore_SP_CC.jks
```

```
keystore/eidasKeyStore_SP_CD.jks
keystore/eidasKeyStore_SP_CF.jks
sp/EncryptModule_SP.xml
sp/saml-engine-additional-attributes.xml
sp/saml-engine-eidas-attributes.xml
sp/SamlEngine_SP.xml
sp/SignModule_SP.xml
sp/sp.properties
sp/SPSamlEngine.xml
specific/eidas_Specific.xml
specific/EncryptModule_SP-Specific.xml
specific/EncryptModule_Specific-IdP.xml
specific/saml-engine-additional-attributes.xml
specific/SamlEngine_SP-Specific.xml
specific/SamlEngine_Specific-IdP.xml
specific/SignModule_SP-Specific.xml
specific/SignModule_Specific-IdP.xml
specific/SpecificSamlEngine.xml
```

9. Copy the server configuration files and test keystores into the local directories defined in step 8.

   Open the zip file (`config.zip` in the `EIDAS-Binaries-xxx-yyy.zip`) and copy the directory `keystore` and the directory of the application server as required (i.e. `glassfish`, `tomcat`, `jboss`, `wls`, `was`) into the configuration directory.

10. On WebSphere Liberty Profile the following features should be enabled:

    ```
    <feature>jsp-2.2</feature>
    <feature>servlet-3.0</feature>
    ```
    `<feature>ssl-1.0</feature>` (if planning to use https)

11. On WebSphere, use `BouncyCastle` provider instead of default IBM JVM default provider:

    a. Place `bouncycastle jar` in `$IBM_JRE\lib\ext` directory.

    b. Copy the IBM unrestricted JCE policy files provided in `AdditionalFiles` directory and put them under `$IBM_JRE\lib\security` to replace the existing ones.

    c. Add `bouncycastleprovider` in the list of providers in `$IBM_JRE\lib\security\java.security` file before the default provider, e.g.

    ```
    security.provider.1=com.ibm.crypto.pkcs11impl.provider.IBMPKCS11Impl
    security.provider.2=org.bouncycastle.jce.provider.BouncyCastleProvider
    security.provider.3=com.ibm.crypto.provider.IBMJCE
    ```

12. Add a static JCE for JBOSS 6/7:

    a. Locate and open in a text editor the file `$JRE_HOME\lib\security\java.security`.

    b. Add a line after the lines containing the security providers:
    `security.provider.N=`
    `org.bouncycastle.jce.provider.BouncyCastleProvider`

(you should set N according to your config, to the next available index in the list of providers).

c. Put bcprov-jdk15on-1.51.jar into the classpath (e.g. $JRE_HOME\lib\ext).

13. Deploy the applications according to your application server.

- `EidasNode.war`
- `SP.war`
- `IdP.war`

You now have a Service Provider, eIDAS-Node Connector, eIDAS-Node Proxy Service and IdP configured to run on localhost:

- Tomcat: `http://localhost:8080/SP/`
- Glassfish: `http://localhost:8081/SP`
- Jboss: `http://localhost:8085/SP`
- WebLogic: `http://localhost:7001/SP`
- WebSphere, WebSphere Liberty Profile: `http://localhost:9080/SP/`

To validate the installation, a first test can be performed simulating that a citizen from a country accesses services in the same country.

1. Open the Service Provider URL : `http://localhost:defaultport/SP/`
2. Choose for both the SP and citizen country the fictitious country for which your application server has been configured (CA, CB, CC, CD or CF).
3. The generated `SAMLRequest` is displayed. Submit the form.
4. Click **Next** to give your consent to attributes being transferred.
5. Enter the user credentials. Type 'xavi' as **Username** and 'creus' as **Password** and submit the page.
6. Click **Submit** to validate the values to transfer.

   The `SAMLResponse` is displayed.
7. **Submit** the form.

   You should see **Login Succeeded**.

 

# 5. Specific configuration

## 5.1. Changing the default hostname or http port

The parameters below can be adapted to reflect your configuration.

**Note:** The application server must be restarted after changes have been made.

### 5.1.1. eIDAS-Node hostname and port

1. Edit the file `eidas.xml` located in the configuration directory as shown below.

| Property | Value |
|---|---|
| `connector.assertion.url` | http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ColleagueResponse |
| `connector.destination.url` | http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ServiceProvider |
| `connector.node.url` | http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ |
| `connector.metadata.url` | http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ConnectorMetadata |
| `service.node.url` | http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/ |
| `service1.url` | http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/ColleagueRequest |
| `service.specificidpredirect.url` | http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/IdpResponse |
| `service.metadata.url` | http://<service.*yourHostname*>:<servicer.*yourPort*>/EidasNode/ServiceMetadata |
| `connector.responder.metadata.url` | The URL at which the metadata of the eIDAS-Node Connector (presenting itself as an IdP) will be made available, e.g. http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ConnectorResponderMetadata. It will be used as Issuer in the responses that the Connector sends to the SP. |
| `service.requester.metadata.url` | The URL where the metadata of the Proxy Service (presenting itself as an SP) will be made available, e.g. http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/ServiceRequesterMetadata . It will be used as Issuer in the requests that the Connector sends to the IdP. |
| `ssos.serviceMetadataGeneratorIDP.post.location` | The URL for the metadata `<md:SingleSignOnService>` location attribute of the `SingleSignOnService` related to `Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST`. e.g. http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/ColleagueRequest/ |

| Property | Value |
|---|---|
| ssos.serviceMetadataGeneratorIDP.redirect.location | The URL for the metadata <md:SingleSignOnService> location attribute of the SingleSignOnService related to Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect. e. g. http://*<service.yourHostname>*:*<service.yourPort>*/EidasNode/ColleagueRequest/ |
| ssos.connectorMetadataGeneratorIDP.post.location | The URL for the metadata <md:SingleSignOnService> location attribute of the SingleSignOnService related to Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST. e.g. http://*<connector.yourHostname>*:*<connector.yourPort>*/EidasNode/ServiceProvider |
| ssos.connectorMetadataGeneratorIDP.redirect.location | The URL for the metadata <md:SingleSignOnService> location attribute of the SingleSignOnService related to Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect. e.g. http://*<connector.yourHostname>*:*<connector.yourPort>*/EidasNode/ServiceProvider |

2. Open and edit the file sp.properties as shown below.

| Property | Value |
|---|---|
| country1.metadata.url | http://<connector.*yourHostname*>/EidasNode/ConnectorResponderMetadata |

### 5.1.2. SP hostname and port

Open and edit the file sp.properties as shown below.

| Property | Value |
|---|---|
| sp.return | http:// <sp.*yourHostname*>:<sp.*yourPort*>/SP/ReturnPage |
| sp.metadata.url | http:// <sp.*yourHostname*>:<sp.*yourPort*>/SP/metadata |

### 5.1.3. IdP hostname and port

1. Edit the file eidas_Specific.xml located in the configuration folder as shown below.

| Property | Value |
|---|---|
| idp.url | http://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/AuthenticateCitizen |
| idp.metadata.url | https://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/metadata |

2. Open and edit the file idp.properties as shown below.

| Property | Value |
|---|---|
| idp.metadata.url | http://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/metadata |
| idp.ssos.redirect.location | The URL for the metadata <md:SingleSignOnService> location attribute of the SingleSignOnService related to Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect. e.g. http://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/AuthenticateCitizen |
| idp.ssos.post.location | The URL for the metadata <md:SingleSignOnService> location attribute of the SingleSignOnService related to Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST. e.g. http://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/AuthenticateCitizen |

## 5.2. Changing the keystore location

By default the test keystores are located in the directory 'keystores' under the configuration directory. You can change these values by editing the files below to reflect your configuration. All filenames and path information are relative to the configuration directory for the given module.

| Keystore | Files |
|---|---|
| **eIDAS-Node** | SignModule_Service.xml<br>SignModule_Connector.xml<br>EncryptModule_Service.xml<br>EncryptModule_Connector.xml |
| **SP** | SignModule_SP.xml<br>EncryptModule_SP.xml |
| **IdP** | SignModule_IdP.xml<br>EncryptModule_IdP.xml |
| **Specific** | SignModule_SP-Specific.xml<br>SignModule_Specific-IdP.xml<br>EncryptModule_SP-Specific.xml<br>EncryptModule_Specific-IdP.xml |

## 5.3. Changing keystore configuration

By default the preconfigured eIDAS components use the following extended configuration.

### 5.3.1. Extended configuration

In this configuration all stakeholders (SP/Connector /Proxy Service/IDP) use their own certificate for the signing and encrypting of SAML messages.

This setup is close to a real-life scenario, where the components are distributed across servers and Member States.

Example for country 'CA':

| | Keystore | | Certificate | Country |
|---|---|---|---|---|
| **SP** | eidasKeyStore_SP_CA.jks<br><br>(SignModule_SP.xml, EncryptModule_SP.xml) | Key Pair | sp-ca-demo-certificate | CA |
| | | Trusted | Metadata (signing certificate) | CA |
| **Connector** | eidasKeyStore_Connector_CA.jks<br><br>(SignModule_SP-Specific.xml, SignModule_Connector.xml, EncryptModule_SP-Specific.xml EncryptModule_Connector.xml) | Key Pair | Connector-ca-demo-certificate | CA |
| | | Trusted | Metadata (signing certificate) | CA |
| **Proxy Service** | eidasKeyStore_Service_CA.jks<br><br>(SignModule_Service.xml, SignModule_Specific-IdP.xml, EncryptModule_Service.xml, EncryptModule_Specific-IdP.xml) | Key Pair | Service-ca-demo-certificate | CA |
| | | Trusted | Metadata (signing certificate) | CA |
| **IDP** | eidasKeyStore_IDP_CA.jks<br><br>(SignModule_IdP.xml, EncryptModule_IdP) | Key Pair | idp-ca-demo-certificate | CA |
| | | Trusted | Metadata (signing certificate) | CA |
| **Metadata** | eidasKeyStore_METADATA.jks | Key Pair | Metadata (signing certificate) | CA |

### 5.3.2. Basic configuration

In this configuration all stakeholders share the same certificate.

This setup is a simplified scenario for a lab environment, but corresponds less to a real-life situation.

In order to set up the basic scenario, all `SignModule` configuration files should be adapted to reference the common test keystore, `eidasKeyStore.jks`.

## 5.4. Preventing a citizen from authenticating in a country other than the requested one

1. By default the preconfigured Demo eIDAS-Node has a protection which does not allow citizens to authenticate in a country other than the one that has been requested.

2. If you need to disable this validation, edit the file eidas.xml located in the configuration directory.

| Property | Value |
|---|---|
| check.citizenCertificate.serviceCertificate | false |

## 5.5. eIDAS-Node compliance

For validation purposes the demo eIDAS Nodes do not use HTTPS and the configuration parameters are set as shown below. The parameters can be changed to be fully eIDAS compliant if required.

| Parameter | Demo value | eIDAS value |
|---|---|---|
| disallow_self_signed_certificate | False | True: do not allow self-signed and expired certificates |
| check_certificate_validity_period | False | True: do not allow expired certificates |
| metadata.activate | True | True:specifies that metadata is generated by the Connector |
| metadata.restrict.http | False | True: metadata must be only available via HTTPS |
| tls.enabled.protocols | TLSv1.1,TLSv1.2 | TLSv1.1,TLSv1.2: SSL/TLS enabled protocols |
| metadata.check.signature | True | True : metadata received from a communications partner must be signed |
| metadata.validity.duration | 86400 | Metadata validity period in seconds. Default=86400 (i.e. one day) |
| response.encryption.mandatory | True | True: do not allow response not encrypted |
| validate.binding | True | True: the bindings are validated |
| security.header.csp.enabled | True | True: the content-security and security checks are enabled |
| disable.check.mandatory.eidas.attributes | False | False: check the eIDAS minimum dataset constraint.<br><br>**Note:** this parameter is used by both Proxy Service and Connector. |
| disable.check.representative.attributes | False | True: disable the check of representative attributes in the request |

## 6. Compiling the modules from the source

If you need to rebuild the Maven project, open EIDAS-Parent and execute the Maven commands described in the table below according to your application server.

| Folder | Command line | |
|---|---|---|
| **EIDAS-Parent** | Tomcat/ Glassfish | `mvn clean install -P tomcat` |
| | jBoss6 | `mvn clean install -P jBoss6` |
| | jBoss7 | `mvn clean install -P jBoss7` |
| | WebLogic | `mvn clean install -P weblogic` |
| | WebSphere | `mvn clean install -P websphere` |

You can also rebuild each module separately by executing the commands below.

| Folder - modules | Command line | |
|---|---|---|
| **EIDAS-Light-Commons** | `mvn clean install` | |
| **EIDAS-Commons** | `mvn clean install` | |
| **Eidas-ConfigModule** | `mvn clean install` | |
| **EIDAS-SpecificCommunicationDefinition** | `mvn clean install` | |
| **EIDAS-SAMLEngine** | `mvn clean install` | |
| **EIDAS-Encryption** | `mvn clean install` | |
| **EIDAS-UPDATER** | `mvn clean install` | |
| **EIDAS-Specific** | `mvn clean install` | |
| **EIDAS-Node** | Tomcat/ Glassfish | `mvn clean package -P tomcat` |
| | jBoss6 | `mvn clean package -P jBoss6` |
| | jBoss7 | `mvn clean package -P jBoss7` |
| | WebLogic | `mvn clean package -P weblogic` |
| | WebSphere | `mvn clean package -P websphere` |

| Folder - modules | Command line | |
|---|---|---|
| **EIDAS-SP** | Tomcat/<br>Glassfish | `mvn clean package -P tomcat` |
| | jBoss6 | `mvn clean package -P jBoss6` |
| | jBoss7 | `mvn clean package -P jBoss7` |
| | WebLogic | `mvn clean package -P weblogic` |
| | WebSphere | `mvn clean package –P websphere` |
| **EIDAS-IdP-1.0** | Tomcat/<br>Glassfish | `mvn clean package –P tomcat` |
| | jBoss6 | `mvn clean package -P jBoss6` |
| | jBoss7 | `mvn clean package -P jBoss7` |
| | WebLogic | `mvn clean package -P weblogic` |
| | WebSphere | `mvn clean package –P websphere` |

## 7. Enabling logging

The locations of the audit files are by default configured to use a Java system properties variable called `LOG_HOME`.

A value can be assigned to this variable by using: `-DLOG_HOME="<myDirectoryName>"` at server start-up.