eIDAS Node Installation, Configuration and Integration

Quick Start Guide

Version 1.2

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 Update of the deployments configuration and related libraries	DIGIT
1.2	14/03/2017	Configuration and stability improvements, please see section 2 — Release content.	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.
MW	Middle Ware. Architecture of the integration of eIDs in Services, with a direct communication between SP and the citizen's PC without any central server. The term also refers to the piece of software of this architecture that executes on the citizen's PC.
MS	Member State.
SAML	Security Assertion Markup Language
SP	Service Provider
STORK	Secure idenTity acrOss boRders linKed

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, will be included in the eIDAS-Node *Installation, Configuration and Integration Manual* (to be published later).

This document is divided into the following sections:

- Section 1 – *Introduction*;
- Section *2* – *Release content* lists the components that are delivered in this package;
- 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

Updates in version 1.2.0 include architectural and stability improvements:

- Improvements in security
  - Penetration tested;
- Improvement in code quality:
  - Corrections based on code quality analysis.
- Improvements in build process:
  - Reorganising Maven POM in a standardised way;
- Improvements in eIDAS-Node configuration:
  - Make eIDAS software compliant with eIDAS specification regarding TLS version by introducing new configuration property `tls.enabled.protocols`;
  - Add configuration properties `service.askconsent.all.attributes`, `service.askconsent.attribute.names.only` to manage the business attribute/Value in the consent page.
- Improvements in metadata:
  - Metadata was double-signed. Both the Entity descriptor as well as the role descriptor were signed. Only the root element is now signed.
- Improvements in Specific module configuration:
  - Merge of the two files `specific.properties` and `eidas_Specific.xml` into `eidas_Specific.xml;`
  - Rename SAML Engine configuration files, `XXX_Specific.xml` is renamed to `XXX_Specific-IdP.xml`, `XXX_SP-Connector.xml` is renamed to `XXX_SP-Specific.xml`.
- Improvements in sample SP configuration:
  - Add configuration properties (`sp.metadata.validatesignature`, `sp.metadata.trusteddescriptors`) to manage the validation of the metadata signature.
- Improvements in sample IDP configuration:
  - Add configuration properties (`idp.metadata.validatesignature`, `idp.metadata.trusteddescriptors`) to manage the validation of the metadata signature.
- Improved utilisation of Hazelcast: now only one Hazelcast instance is used by default, but it can be reconfigured to have multiple instances in application context.

Deliverable	Description
`EIDAS-1.2.0.zip`	Distribution version 1.1.0 of the sample eIDAS-Node

Deliverable	Description
`EIDAS-Sources-1.2.0.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-1.2.0.zip`	Deployable war files of a preconfigured eIDAS-Node for a Glassfish server (including IdP.war, EidasNode.war, SP.war)
`EIDAS-Binaries-Jboss-1.2.0.zip`	Deployable war files of a preconfigured eIDAS-Node for a JBoss server (including IdP.war, EidasNode.war, SP.war)
`EIDAS-Binaries-Tomcat-1.2.0.zip`	Deployable war files of a preconfigured eIDAS-Node for a Tomcat server (including IdP.war, EidasNode.war, SP.war)
`EIDAS-Binaries-Was-1.2.0.zip`	Deployable war files of a preconfigured eIDAS-Node for a WebSphere server (including IdP.war, EidasNode.war, SP.war)
`EIDAS-Binaries-Wls-1.2.0.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/ **WebSphere Liberty Profile**	8.5.5* 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 *5.2 — 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. A local directory must be defined in order to store the configuration files and the test keystores. We refer to this directory as `<LOCALCONF>`. By default, it is set to `C:\Pgm\projects\configEidas` in the file `environmentContext.xml`.

   If you need to change this location, refer to section *5.1 — Changing the configuration directory <LOCALCONF>* for details.

9. Copy the server configuration files and test keystores into the local directory `<LOCALCONF>`.

   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 `<LOCALCONF>` directory.

10. If you have a different `<LOCALCONF>` than the default one you will also need to update the file `<LOCALCONF>/tomcat/eidas.xml` file with the new `<LOCALCONF>` in the following three lines:

```
<entry
key="metadata.file.repository">c:\Pgm\projects\configEidas\tomcat\meta
data
</entry>
```

11. 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)
```

12. 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
```

13. 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).

14. 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 configuration directory <LOCALCONF>

If you need to change the location of the configuration directory, open the `EidasNode.war` file and edit the `environmentContext.xml` file to reflect your configuration.

Remark: If you change the `localConf` path, please refer to section *5.3 — Changing the keystore location* in order to be consistent in the changes.

File: `\EidasNode\WEB-INF\classes\environmentContext.xml`

```
<!–
    // Defining the eidasConfigRepository using an environment variable
    <bean id="eidasConfigRepository" class="java.lang.String">
        <constructor-arg value="#{systemEnvironment['EIDAS_CONFIG_REPOSITORY']}" />
    </bean>
-->
<!--
    // Defining the eidasConfigRepository using a System property
    <bean id="eidasConfigRepository" class="java.lang.String">
        <constructor-arg value="#{systemProperties['EIDAS_CONFIG_REPOSITORY']}" />
    </bean>
-->
<!-- Defining the eidasConfigRepository using a plain String -->
<bean id="eidasConfigRepository" class="java.lang.String">
   <constructor-arg value="c:/Pgm/projects/configEidas/jboss/"/>
</bean>
```

5.2. 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.

5.2.1. eIDAS-Node hostname and port

1. Edit the file `eidas.xml` located in the `<LOCALCONF>` 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.metadata.url`	http://<connector.*yourHostname*>:<connector.*yourPort*>/EidasNode/ConnectorMetadata
`service1.url`	http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/ColleagueRequest

Property	Value
`service.specificidpredirect.url`	http://<service.*yourHostname*>:<service.*yourPort*>/EidasNode/IdpResponse
`service.metadata.url`	http://<connector.*yourHostname*>:<connector.*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/
`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 the `SP.war` and edit the file `\SP\WEB-INF\classes\sp.properties` as shown below.

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

5.2.2. SP hostname and port

Open the SP.war and edit the file \SP\WEB-INF\classes\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

5.2.3. 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 the IdP.war and edit the file \IdP\WEB-INF\classes\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

5.3. Changing the keystore location

By default the test keystores are located in the directory <LOCALCONF>. You can change these values by editing the files below to reflect your configuration.

Keystore	Files
eIDAS-Node	\EidasNode\WEB-INF\classes\SignModule_Service.xml \EidasNode\WEB-INF\classes\SignModule_Connector_Service.xml \EidasNode\WEB-INF\classes\EncryptModule_Service.xml \EidasNode\WEB-INF\classes\EncryptModule_Connector_Service.xml
SP	\SP\WEB-INF\classes\SignModule_SP.xml \SP\WEB-INF\classes\EncryptModule_SP.xml
IdP	\IdP\WEB-INF\classes\SignModule_IdP.xml

	`\IdP\WEB-INF\classes\EncryptModule_IdP.xml`

Specific	Open the `EidasNode.war`, then open the jar file
	`\EidasNode\WEB-INF\lib\eidas-specific.1.1.0.jar`
	and edit the files:
	`SignModule_SP-Specific.xml`
	`SignModule_Specific-IdP.xml`
	`EncryptModule_SP-Specific.xml`
	`EncryptModule_Specific-IdP.xml`

5.4. Changing keystore configuration

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

5.4.1. 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` (`SignModule_SP.xml`, `EncryptModule_SP.xml`)	Key Pair	sp-ca-demo-certificate	CA
		Trusted	Metadata (signing certificate)	CA
Connector	`eidasKeyStore_Connector_CA.jks` (`SignModule_SP-Specific.xml`, `SignModule_Connector-Service.xml`, `EncryptModule_SP-Specific.xml` `EncryptModule_Connector_Service.xml`)	Key Pair	Connector-ca-demo-certificate	CA
		Trusted	Metadata (signing certificate)	CA
Proxy Service	`eidasKeyStore_Service_CA.jks` (`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` (`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

5.4.2. 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`.

5.5. 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 `<LOCALCONF>` directory.

Property	Value
`check.citizenCertificate.serviceCertificate`	false

5.6. 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.https`	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. **Note:** this parameter is used by both Proxy Service and Connector.

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-SpecificCommun icationDefinition	mvn clean install	
EIDAS-CertManagement	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/ 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/ 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`

If you plan to change the `<LOCALCONF>` directory from the default value, the following files should be edited to reflect your `<LOCALCONF>` value:

```
EIDAS-Sources-1.1.0\EIDAS-Specific\src\main\config\embedded\SignModule_Specific-
IdP.xml
EIDAS-Sources-1.1.0\EIDAS-Specific\src\main\config\embedded\SignModule_SP-
Specific.xml
EIDAS-Sources-1.1.0\EIDAS-NODE\src\main\resources\environmentContext.xml
EIDAS-Sources-1.1.0\EIDAS-NODE\src\main\config\embedded\SignModule_Service.xml
EIDAS-Sources-1.1.0\EIDAS-NODE\src\main\config\embedded\SignModule_Connector-
Service.xml
EIDAS-Sources-1.1.0\EIDAS-SAMLEngine\src\test\resources\SignModule_Conf1.xml
EIDAS-Sources-1.1.0\EIDAS-SAMLEngine\src\test\resources\SignModule_Conf2.xml
EIDAS-Sources-1.1.0\EIDAS-SAMLEngine\src\test\resources\SignModule_Conf3.xml
EIDAS-Sources-1.1.0\EIDAS-SP\src\main\resources\SignModule_SP.xml
EIDAS-Sources-1.1.0\EIDAS-Specific\src\main\config\embedded\EncryptModule_Specific-
IdP.xml
EIDAS-Sources-1.1.0\EIDAS-Specific\src\main\config\embedded\EncryptModule_SP-
Specific.xml
EIDAS-Sources-1.1.0\EIDAS-NODE\src\main\config\embedded\EncryptModule_Service.xml
EIDAS-Sources-1.1.0\EIDAS-NODE\src\main\config\embedded\EncryptModule_Connector-
Service.xml
EIDAS-Sources-1.1.0\EIDAS-SAMLEngine\src\test\resources\EncryptModule_SP.xml
EIDAS-Sources-1.1.0\EIDAS-SAMLEngine\src\test\resources\EncryptModule_IdP.xml
EIDAS-Sources-1.1.0\EIDAS-IdP-1.0\src\main\config\embedded\SignModule_IdP.xml
EIDAS-Sources-1.1.0\EIDAS-IdP-1.0\src\main\config\embedded\EncryptModule_IdP.xml
```

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.