eIDAS-Node Installation, Configuration and Integration Manual

Version 1.3

Document history

Version	Date	Reason for modification	Modified by
1.0	26/11/2015	Modifications to align with the eIDAS technical specifications.	DIGIT
1.1	09/09/2016	• Configuration improvements including support for Tomcat 8. • Removal of Attribute Provider. • Documentation of improvements included in Release 1.1 (see Release notes for eIDAS-Node version 1.1).	DIGIT
1.2	20/01/2017	• Configuration and stability improvements. • Documentation of improvements included in Release 1.2.0 (see Release notes for eIDAS-Node version 1.2.0).	DIGIT
1.3	08/06/2017	• Modifications to align with changes in Technical Specifications version 1.1. • Bug fixes and configuration improvements (for details please see the Version 1.3.0 Release Notes). • Documentation improvements to remove eIDAS-Nodes error codes and place in separate document *eIDAS Error Codes*.	DIGIT

Disclaimer

This document is for informational purposes only and the Commission cannot be held responsible for any use which may be made of the information contained therein. References to legal acts or documentation of the European Union (EU) cannot be perceived as amending legislation in force or other EU documentation.

The document contains a brief overview of technical nature and is not supplementing or amending terms and conditions of any procurement procedure; therefore, no compensation claim can be based on the contents of the present document.

Table of contents

List of figures

List of tables

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
PEPS	Pan European Proxy Service (as defined in the STORK Pilots). Now known as eIDAS-Node.
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
Audit	A function which seeks to validate that controls are in place, adequate for their purposes, and which reports inadequacies to appropriate levels of management.
Audit log	An audit log is a chronological sequence of audit records, each of which contains evidence directly as a result of the execution of a business process or system function
Audit trail	A chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities surrounding or leading to an operation, procedure, or event in a security-relevant transaction from inception to final results.
Auditable event	An auditable event is generated by any activity in a system that is capable of being audited. An auditable event could lead to the compromise of the integrity and/or security of an information system and therefore indirectly compromise a business process
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. This was formerly known as S-PEPS.
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. This was formerly known as C-PEPS.
Encryption	Any procedure used in cryptography to convert plain text into cipher text in order to prevent anyone but the intended recipient from reading that data.
Hashing	The process of using a mathematical algorithm against data to produce a numeric value that is representative of that data.

Security event	See Auditable event

References

[1] ISO/IEC 27002 - Information technology -- Security techniques -- Code of practice for information security management, section 10.10, 2005 (www.iso.org)

[2] BSI PD008: Legal Admissibility and Evidential Weight of Information Stored Electronically, British Standards Institution, 1999

[3] COBIT (Control Objectives for Information and related Technology) from Information Systems Audit and Control Association (http://www.isaca.org/cobit.htm)

[4] ICT-PSP/2007/1 – STORK 1 : D5.7.3 Functional Design for PEPS, MW models and interoperability

[5] K. Kent, M. Souppaya. Guide to Computer Security Log Management. Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-92, September 2006

[6] SANS Consensus Policy Resource Community - Information Logging Standard, http://www.sans.org/security-resources/policies/server-security

[7] NIST: An Introduction to Computer Security: The NIST Handbook, NIST Special Publication 800-12, December 1997, http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf

[8] Common Criteria: Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 4, September.2012 Part 2: Security Functional Components, http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf

[9] ENISA: Privacy Features of European eID Card Specification, Version 1.0.1, January 2009, http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_privacy_features_eID.pdf

[10] EC council - The use of audit trails in security systems http://www.europeanpaymentscouncil.eu/index.cfm/knowledge-bank/epc-documents/the-use-of-audit-trails-in-security-systems-guidelines-for-european-banks/

[11] AUDIT Trails - NIST publication http://csrc.nist.gov/publications/nistbul/itl97-03.txt

1. Introduction

This document is intended for a technical audience consisting of developers, administrators and those requiring detailed technical information on how to configure, build and deploy the eIDAS-Node application.

The document describes the steps involved when implementing a basic setup and goes on to provide detailed information required for customisation and deployment.

1.1. Document structure

This document is divided into the following sections:

- Chapter 1 – *Introduction*: this section.
- Chapter 2 – *Before Starting* describes the steps to follow to configure an Application Server (Tomcat, JBoss, etc.) to serve as a container for your application.
- Chapter 3 – Basic *Set up* describes how to quickly set up, compile and run one instance of your Application Server.
- Chapter 4 – *Detailed Set up — Installing* provides detailed descriptions of the configurations to enable you to change specific aspects as required.
- Chapter 5 – *Server Final File Structure* shows the final structure of your application server relevant directories, so that you can confirm that you have made the proper configurations.
- Chapter 6 – *Developing Specific Parts* provides information for the development of each country's specific parts.
- Chapter 7 — *eIDAS-Node compliance* describes the production mode for ensuring eIDAS regulation compliance.
- Chapter 8 — *eIDAS Node SAML XML Encryption* describes the W3C recommendations and how SAML XML encryption is implemented and integrated in eID.
- Chapter 9 – *eIDAS-Node SAML metadata* describes the implementation of SAML metadata in eID.
- Chapter 10 — *eIDAS-Node SAML Engine* provides information about the SAML engine which is used by any module that needs to generate or validate SAML messages with the eIDAS specification.
- Chapter 11 – *Support for additional attributes* describes how to provide support for attributes in addition to the eIDAS minimum dataset.
- Chapter 12 – *ProtocolEngine Configuration* describes configuring and customisation of the new ProtocolEngine.
- Chapter 13 – *eIDAS-Node Security Headers* describes the HTTP response headers the eIDAS-Node can send in order to increase its security.
- Chapter 14 — *Clustering Environment* describes the technologies and configurations used for testing the eIDAS-Node in cluster mode. The choice of technologies is proposed for testing purpose.
- Chapter 15 — *eIDAS-Node Error and Event Logging* provides information on the eID implementation of error and event logging as a building block for generating an audit trail of activity on the eIDAS Network.
- Chapter 16 — Specific interface

- Appendix A – *eIDAS-Node Error Codes and Error Messages* provides a list of eIDAS-Node errors and related error messages.
- Appendix B – *eIDAS Levels of Assurance* provides information on the three Levels of Assurance described in the Implementing Regulation.
- Appendix C – *User Consent* provides a brief overview of the meaning of 'user consent' in the context of privacy legislation.
- Appendix D — *Hazelcast proposed configuration* provides specific information related to configuration of a cluster environment using Hazelcast.
- Appendix E — *Installation Frequently Asked Questions* provides answers to questions that may arise during your installation.

1.2. Document aims

The aims of this document are to:

- guide you through the preliminary steps involved when setting up your servers;
- guide you through setting up, compiling and running a project for a basic configuration with one instance of your Application Server;
- cover detailed configuration of eIDAS-Nodes;
- provide a check list of files for each application server;
- provide details of how to develop and tailor the Specific parts for your country;
- show how to ensure eIDAS regulation compliance and provide a check list of recommendations;
- describe the W3C encryption recommendations and show how SAML XML encryption is implemented and integrated in eID;
- describe the implementation of SAML metadata in eID;
- provide information about how SAML messages are generated and validated by the SAML Engine;
- describe the HTTP response headers that the eIDAS-Node can send in order to increase its security;
- describe the technologies and configurations used for testing the eIDAS-Node in cluster mode;
- provide information on the eID implementation of error and event logging as a building block for generating an audit trail of activity.

1.3. eIDAS Technical specifications and software provided

This software package is provided as a sample implementation in accordance with the *eIDAS Technical Specifications* v1.1.

1.3.1. Further information

For further information on the practical implementation of the features listed above, please refer to the relevant chapter:

- Chapter 7 — *eIDAS-Node compliance* describes the production mode for ensuring eIDAS regulation compliance.

- Chapter 8 — *eIDAS Node SAML XML Encryption* describes the W3C recommendations and how SAML XML encryption is implemented and integrated in eID.
- Chapter 9 – *eIDAS-Node SAML metadata* describes the implementation of SAML metadata in eID.
- Chapter 10 — *eIDAS-Node SAML Engine* provides information about the SAML engine which is used by any module that needs to generate or validate SAML messages with the eIDAS specification.

2. Before Starting

The project build files are in **Maven3** format, so you need to install Maven. Download instructions are provided at http://maven.apache.org/run-maven/index.html).
Recommended versions of Maven are 3.2.1 and above. Lower versions can result in an exception when building, e.g. `java.lang.NoClassDefFoundError`.

The project is built by default using the **Java SDK** version **1.7** (and can also be built in Java 1.8).

In order to avoid a possible XML External Entity attack (XXE), the OWASP guidelines advise to use Java 7 update 67, Java 8 update 20 or above. For more details, please refer to
https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Prevention_Cheat_Sheet.

The following is a list of the supported servers.

Table 1: Supported servers

Application Server	Supported version(s)
Tomcat	7, 8
GlassFish	3, 4 Web Profile
JBoss	6, 7
WebLogic	10.3.6, 12.1.2
WebSphere/WebSphere Liberty Profile	8.5.5/8.5.5.4

Instructions on how to deploy the project on Tomcat or JBoss, GlassFish, WebLogic and WebSphere servers are provided in section *2.2 — Configuring the Server*.

The following variables are used during the installation process:

- `$TOMCAT_HOME` – Base directory of your Tomcat server. (e.g. `\home\user\apps\apache-tomcat-7.0.62`);
- `$JBOSS_HOME` – Base directory of your JBoss server. (e.g. `\home\user\apps\jboss-7.4.0`);
- `$SERVER_CONFIG` – JBoss server configuration name (e.g. default)

If you want to use the 'default' configuration server, your full path will be: `\home\user\apps\jboss-7.4.0.GA\server\default`;

- `$GLASSFISH_HOME` – Base directory of your GlassFish server (e.g. `\home\user\apps\glassfishv3`).
- `$EIDAS_CONFIG_REPOSITORY` – used in `applicationContext.xml` and points to the configuration directory of the application server (e.g. `file:/C:/PGM/projects/configEidas/jboss/`).

2.1. Setting up your Keystore

Copy your `eidasKeystore.jks` (the key store with your eIDAS-Node keys, alternatively you can use the example key store provided with the application) into a directory of your own choice, and make sure that:

- the property `keyStorePath` on file: `$EIDAS_CONFIG_REPOSITORY/SignModule_Service.xml` reflects the relative location of your Proxy Service `eidasKeyStore.jks`;
- the property `keyStorePath` on file: `$EIDAS_CONFIG_REPOSITORY/SignModule_Connector.xml` reflects the relative location of your eIDAS-Node Connector `eidasKeyStore.jks`;
- the property `keyStorePath` on file: `$SPECIFIC_CONFIG_REPOSITORY/SignModule_SP-Specific.xml` reflects the relative location of your eIDAS-Node Connector to national infrastructure specific `eidasKeyStore.jks`;
- the property `keyStorePath` on files: `$SP_CONFIG_REPOSITORY/SignModule_SP.xml` reflects the relative location of your SP `eidasKeyStore.jks`;
- the property `keyStorePath` on files: `$IDP_CONFIG_REPOSITORY/SignModule_IdP.xml` reflects the relative location of your IdP `eidasKeyStore.jks`;
- the property keyStorePath on files: `$SPECIFIC_CONFIG_REPOSITORY`/SignModule_Specific-IdP.xml reflects the relative location of your Proxy Service to your national infrastructure specific `eidasKeyStore.jks`.

If the eIDAS-Node is configured to use encryption, also ensure that:

- the property `keyStorePath` on file: `$EIDAS_CONFIG_REPOSITORY/EncryptModule_Service.xml` reflects the relative location of your Proxy Service `eidasKeyStore.jks`;
- the property `keyStorePath` on file: `$EIDAS_CONFIG_REPOSITORY/EncryptModule_Connector.xml` reflects the relative location of your eIDAS-Node Connector `eidasKeyStore.jks`;
- the property `keyStorePath` on file: `$SPECIFIC_CONFIG_REPOSITORY/EncryptModule_SP_Connector.xml` reflects the relative location of your eIDAS-Node Connector your to national infrastructure specific `eidasKeystore.jks`;
- the property `keyStorePath` on files: `$SP_CONFIG_REPOSITORY/EncryptModule_SP.xml` reflects the relative location of your SP `eidasKeyStore.jks`;
- the property `keyStorePath` on files: `$IDP_CONFIG_REPOSITORY/EncryptModule_IdP.xml` reflects the relative location of your IdP `eidasKeyStore.jks`;
- the property keyStorePath on files: `$SPECIFIC_CONFIG_REPOSITORY`/EncryptModule_Specific-IdP.xml reflects the relative location of your Specific IdP to your national infrastructure specific `eidasKeyStore.jks`.

For more information see section *10 — eIDAS-Node SAML Engine* and section 8 — *eIDAS Node SAML XML Encryption*

2.2. Configuring the Server

This section provides instructions on how to deploy the project on:

- Tomcat 7 and 8;
- JBoss 6 and 7;
- GlassFish v3,v4;
- WebLogic 10.3.6, 12.1.2;
- WebSphere Application Server v. 8.5.5.

2.2.1. Installing Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files

Before starting to configure each server, apply the JCE Unlimited Strength Jurisdiction Policy Files which contain no restriction on cryptographic strengths:

1. 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
2. Uncompress and extract the downloaded zip file (it contains README.txt and two jar files).
3. For the installation, please follow the instructions in the README.txt file.

2.2.2. Configuring Tomcat 7/Tomcat 8

1. Create a folder named endorsed in $TOMCAT_HOME.
2. Create a folder named shared in $TOMCAT_HOME.
3. 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.
4. Extract from the binary zip file (under AdditionalFiles\endorsed) the following libs to $TOMCAT_HOME\shared\lib:

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

2.2.3. Configuring JBoss 6

1. Extract from the binary zip file (under AdditionalFiles\endorsed) the following libs to $JBOSS_HOME\lib\endorsed

```
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
```

The encryption provider `bcprov-jdk15on-1.51.jar` should be loaded as a JCE provider by the JRE used to run the server. You may use one of the following two methods for doing this (compatible with both JBoss6 and JBoss7):

2. Add a static JCE
    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).
3. Make BouncyCastleProvider classes available under a file:\\ protocol.
    a. Add/change JBOSS_CLASSPATH environment option, adding the full path of bcprov-jdk15on-1.51.jar file, e.g. set JBOSS_CLASSPATH=C:\jboss-6.1.0.Final\bcprov-jdk15on-1.51.jar _

    This setting should be available to the JBoss starting script.

2.2.4. Configuring JBoss AS 7

Install the modules found under `AdditionalFile\JBOSS7`. These modules contain BouncyCastle JCE provider and xml-apis. They should be copied under `$JBOSS_HOME\modules` directory.

2.2.5. Configuring GlassFish V3

Extract from the binary zip file (under `AdditionalFiles\endorsed`) the following libs to `$GLASSFISH_HOME\glassfish\lib\endorsed`

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

2.2.6. Configuring GlassFish V4

Under `$GLASSFISH_DOMAIN\lib\ext\` copy `xml-apis-1.4.01.jar`

2.2.7. Configuring WebLogic

Under `$DOMAIN_HOME\lib\` copy `xml-apis-1.4.01.jar`

2.2.8. Configuring WebSphere

The web applications should be deployed using the WAS Admin Console.

2.2.8.1. Configuring IBM SDK Java version

IBM WebSphere Application Server V8.5 comes by default with IBM SDK Java 6. Using IBM Installation Manager, you can install IBM SDK Java 7 as an optional feature. SDK Java 7 can be added at any time to the WAS installation by following the IBM installation procedure described at http://www-01.ibm.com/support/knowledgecenter/SSEQTP_8.5.5/com.ibm.websphere.installation.base.doc/ae/tins_installation_jdk7_gui.html.

Once this is complete, both IBM SDK Java versions 6 and 7 will coexist. To switch the SDK used by server profiles, you can use the `managesdk` command described at http://www-01.ibm.com/support/knowledgecenter/SSAW57_8.5.5/com.ibm.websphere.nd.multiplatform.doc/ae/rxml_managesdk.html.

2.2.8.2. Configuring encryption support

IBM WebSphere runs with IBM Java virtual machine. The default security provider bundled with JVM does not support the default encryption algorithm used by eIDAS (http://www.w3.org/2009/xmlenc11#aes256-gcm). One option is to use BouncyCastleProvider instead of default IBM JVM default provider:

1. Place the bouncycastle jar in $IBM_JRE\lib\ext directory.
2. Copy the IBM unrestricted JCE policy files provided in AdditionalFiles directory and put them under $IBM_JRE\lib\security to erase the existing ones. **Note that those jars are signed**.
3. Add BouncyCastleProvider to the list of providers in the $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
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.security.cmskeystore.CMSProvider
security.provider.8=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
security.provider.9=com.ibm.security.sasl.IBMSASL
security.provider.10=com.ibm.xml.crypto.IBMXMLCryptoProvider
security.provider.11=com.ibm.xml.enc.IBMXMLEncProvider
security.provider.12=org.apache.harmony.security.provider.PolicyProvider
```

2.2.9. Configuring WebSphere Liberty Profile

The `xml-apis-1.4.01.jar` file should be placed under `$SERVER_HOME\lib\global`.

The application may be deployed by copying the war files under `$SERVER_HOME\dropins` directory.

The IBM Installation Manager can be used to install the IBM SDK Java 7 for Liberty Profile (please refer to the IBM official documentation at: http://www.ibm.com/support/knowledgecenter/SSD28V_8.5.5/com.ibm.websphere.wlp.core.doc/ae/twlp_ins_installation_jdk7.html).

2.3. Enabling logging

To enable audit logging of the communications between eIDAS-Node Proxy Service and eIDAS-Node Connector, you should make the following configuration changes. This is part of EIDAS Audit log, for further information, please refer to section *15 — eIDAS-Node Error and Event 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.

If modification of the environment variable is not possible, the value of this variable could also be assigned by adding the following line in the logback.xml file
`<property name="LOG_HOME" value ="<myDirectoryName>" />`

2.3.1. Configuring Audit Logging

Edit the project eIDAS-Node file: `logback.xml` (located in the resource directory) and add the following lines:

```
<?xml version="1.0" encoding="UTF-8" ?>

<!--
        NOTE :
            the environment variable LOG_HOME could be set to indicate the directory
containing the log files
            the log configuration files will be scanned periodically each 30 minutes
            LOG level is defined as below :
                Default level : INFO
                    Console appender (STDOUT)    : inherits from default
                    eIDASNodeDetail appender         : INFO
                    eIDASNodeSystem appender         : INFO
                    eIDASNodeSecurity appender       : INFO
→-->

<configuration scan="true" scanPeriod="30 minutes">

    <!--
        This define the CONSOLE appender - the level of the console appender is based on
the root level
    -->
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
        <encoder>
```

```
            <pattern>%d{yyyy-MM-dd; HH:mm:ss.SSS} [%thread] %-5level %logger{66} %marker -
%X{sessionId} -%X{remoteHost} -%msg%n</pattern>
        </encoder>
    </appender>

    <!--
        This define the FULL Detailed log file appender - the level of the console
appender is INFO by default
    -->
    <appender name="eIDASNodeDetail"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/eIDASNodeDetail.log</file>

        <filter class="ch.qos.logback.classic.filter.ThresholdFilter">
            <level>INFO</level>
        </filter>
        <encoder class="eu.eidas.node.logging.logback_integrity.HashPatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd; HH:mm:ss.SSS} [%thread] %-5level %logger{66} %marker -
%X{sessionId} -%X{remoteHost} -%msg%n</pattern>
        </encoder>
        <param name="Append" value="true" />
        <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
            <maxFileSize>500KB</maxFileSize>
        </triggeringPolicy>
        <!-- Support multiple-JVM writing to the same log file -->
        <prudent>true</prudent>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG_HOME}/eIDASNodeDetail.%d{yyyy-MM-
dd}.log</fileNamePattern>
            <maxHistory>14</maxHistory>
        </rollingPolicy>
    </appender>

    <!--
        This define the SYSTEM Detailed log file appender - the default Filter is
inherited from root level
    -->
    <appender name="eIDASNodeSystem"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/eIDASNodeSystem.log</file>

        <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
            <evaluator class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">
                <marker>SYSTEM</marker>
            </evaluator>
            <onMismatch>DENY</onMismatch>
            <onMatch>ACCEPT</onMatch>
        </filter>
        <encoder class="eu.eidas.node.logging.logback_integrity.HashPatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd; HH:mm:ss.SSS} [%thread] %-5level %logger{66} %marker -
%X{sessionId} -%X{remoteHost} -%msg%n</pattern>
        </encoder>
        <param name="Append" value="true" />
        <!-- Support multiple-JVM writing to the same log file -->
        <prudent>true</prudent>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG HOME}/eIDASNodeSystem.%d{yyyy-MM-
dd}.log</fileNamePattern>
            <maxHistory>14</maxHistory>
```

```
        </rollingPolicy>
    </appender>

    <!--
        This define the SECURITY Detailed log file appender - the default Filter is
inherited from root level
    -->
    <appender name="eIDASNodeSecurity"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/eIDASNodeSecurity.log</file>

        <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
            <evaluator class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">
                <marker>SECURITY_SUCCESS</marker>
                <marker>SECURITY_WARNING</marker>
                <marker>SECURITY_FAILURE</marker>
            </evaluator>
            <onMismatch>DENY</onMismatch>
            <onMatch>ACCEPT</onMatch>
        </filter>
        <encoder class="eu.eidas.node.logging.logback_integrity.HashPatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd; HH:mm:ss.SSS} [%thread] %-5level %logger{66} %marker -
%X{sessionId} -%X{remoteHost} -%msg%n</pattern>
        </encoder>
        <param name="Append" value="true" />
        <!-- Support multiple-JVM writing to the same log file -->
        <prudent>true</prudent>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG HOME}/eIDASNodeSecurity.%d{yyyy-MM-
dd}.log</fileNamePattern>
            <maxHistory>14</maxHistory>
        </rollingPolicy>
    </appender>

    <!--
        This define the SAML exchange Detailed log file appender - the default Filter is
inherited from root level
    -->
    <appender name="eIDASNodeSAMLExchange"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/eIDASNodeSAMLExchange.log</file>

        <filter class="ch.qos.logback.core.filter.EvaluatorFilter">
            <evaluator class="ch.qos.logback.classic.boolex.OnMarkerEvaluator">
                <marker>SAML_EXCHANGE</marker>
            </evaluator>
            <onMismatch>DENY</onMismatch>
            <onMatch>ACCEPT</onMatch>
        </filter>
        <encoder class="eu.eidas.node.logging.logback_integrity.HashPatternLayoutEncoder">
            <pattern>%d{yyyy-MM-dd; HH:mm:ss.SSS} [%thread] %-5level %logger{66} %marker -
%X{sessionId} -%X{remoteHost} -%msg%n</pattern>
        </encoder>
        <param name="Append" value="true" />
        <!-- Support multiple-JVM writing to the same log file -->
        <prudent>true</prudent>
        <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <fileNamePattern>${LOG HOME}/eIDASNodeSAMLExchange.%d{yyyy-MM-
dd}.log</fileNamePattern>
```

```
            <maxHistory>14</maxHistory>
        </rollingPolicy>
    </appender>

    <!--
        This define the API fine grained level
    -->
    <logger name="org.opensaml">
        <level value="ERROR" />
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>
    <logger name="com.opensymphony.xwork2">
        <level value="WARN"/>
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>
    <logger name=" org.apache.struts2">
        <level value="WARN"/>
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>
    <logger name="org.springframework">
        <level value="WARN" />
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>
    <logger name="org.apache.xml.security">
        <level value="WARN" />
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>

    <logger name="eu.eidas.communication.requests">
        <level value="info" />
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>

    <logger name="eu.eidas.communication.responses">
        <level value="info" />
        <appender-ref ref="STDOUT"/>
        <appender-ref ref="eIDASNodeDetail"/>
    </logger>

    <!--
        The root level is set to debug for development purposes, for production
environment it could be set to INFO
    -->
    <root level="DEBUG">
        <appender-ref ref="STDOUT" />
        <appender-ref ref="eIDASNodeSystem" />
        <appender-ref ref="eIDASNodeSecurity" />
        <appender-ref ref="eIDASNodeDetail" />
        <appender-ref ref="eIDASNodeSAMLExchange" />
    </root>
</configuration>
```

Notes:

- The root level of logging defines the detail of logged events, for testing and development purposes, this level should be set to DEBUG; in production environment, it should be INFO.
- Four different log files are generated by the application, depending on the context of the event to log (please refer to section *15 — eIDAS-Node Error and Event Logging* for more details)
  - the Application System log (eIDASNodeSystem),
  - the Application Security log (eIDASNodeSecurity),
  - the Message Exchange log (eIDASNodeSAMLExchange),
  - the Application Detailed log (eIDASNodeDetail)
- `${FILENAME_FULL_PATH}` is the location of the file which will contain the logs. (e.g.: `\opt\eidaslogs\eIDASNodeDetail.log`)

2.4. Configuring Application Server Security

2.4.1. Security Constraints for WebSphere

WebSphere AS is configured by default to not observe security constraints in web applications. To enforce these constraints WebSphere should be configured as shown below.

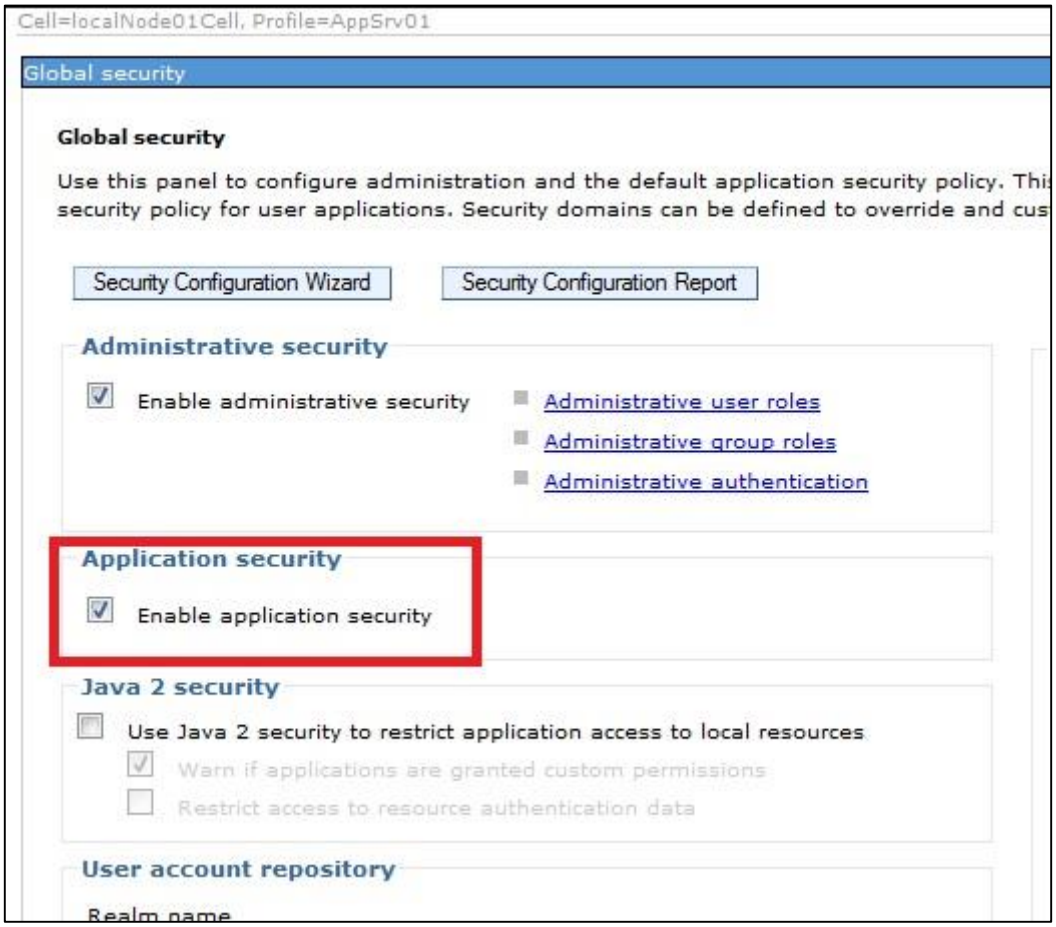


Figure 1: Enabling application security on WebSphere AS

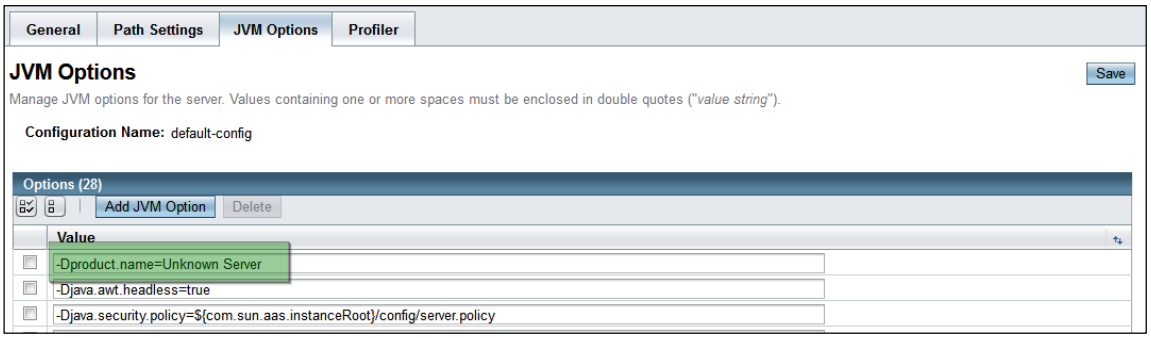
2.5. Setting Software Security Protection

Web server fingerprinting is a critical task to find vulnerabilities. Knowing the version and type of a running web server allows external users to determine known vulnerabilities and the appropriate exploits to use. This information could be useful for someone to prepare an attack against software packages (Servlet and JavaServer Pages) used by the application. This information should be hidden as described below.

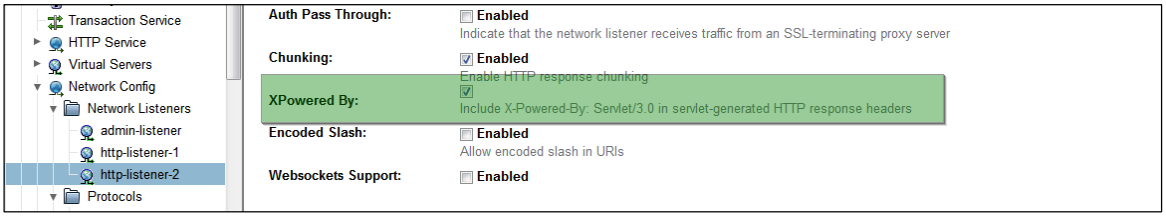
Note: The JavaServer Page `buildVersion.jsp` displays information about the eIDAS software version in use. This has been added for debugging and bug reporting purposes and is valuable for conformance testing. You must remove this before your eIDAS-Node goes live.

2.5.1. GlassFish

1. Open the **JVM Options** tab.
2. Set -Dproduct.name="My App Server"



3. Open the **HTTP** tab.
4. Remove the **XPowered By** flag.



2.5.2. WebLogic

1. In the WebLogic admin console, in **domainName > Configuration > Web Application**, set **X-Powered-By** header to **X-Powered-By** header will **NOT** be sent.
2. Ensure that Send Server Header is disabled In the **Advanced Options** section of the **Server > ServerName > Protocols > HTTP** tab.

2.5.3. JBoss

2.5.3.1. Suppress the X-Powered-By Header JBoss 6, 7 or 7.1

1. To suppress the **X-Powered-By** header in JBoss 6, 7, or 7.1, edit the file catalina.properties located in ${jboss.home}\server\${server.instance.name}\deploy\jbossweb.sar\ and set the property org.apache.catalina.connector.X_POWERED_BY to **false**.
2. Restart the server.

2.5.3.2. Suppress Server information header

To `suppress` the server information header on JBoss servers, add to the `JAVA_OPTS` variable:

```
-Dorg.apache.coyote.http11.Http11Protocol.SERVER=SecureServer
```

2.5.4. Tomcat

2.5.4.1. Modifying the Server Header

To modify the Server Header, in the `${tomcat.home}\conf\server.xml` add a '`server`' attribute and set it as required. The `server` attribute should be set for any HTTP or SSL connectors that you have running.

```
<Connector port="8080" maxHttpHeaderSize="8192"
     maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
     enableLookups="false" redirectPort="8443" acceptCount="100"
     connectionTimeout="20000" disableUploadTimeout="true"
     xpoweredby="true"
             server="SecureServer"/>
```

2.5.5. WebSphere

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console. The following is a list of some of the available web container custom properties.

To specify web container custom properties:

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container**.
2. Under **Additional Properties** select **Custom Properties**.
3. On the **Custom Properties** page, click **New**.
4. On the settings page, enter the name of the custom property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

Note: On WebSphere Application Server v7 you should set the web container's custom property named `com.ibm.ws.webcontainer.invokefilterscompatibility` to true.

2.5.5.1. Suppress the X-Powered-By Header

When you configure server security, you can turn off the `X-Powered-By` header if you do not want to reveal which server you are running. Use this custom property to disable the `X-Powered-By` header, which prevents the header from being sent on the HTTP response. , Set this property to 'true', if you want to disable this header. The default value is 'false'.

3. Basic Set up

This section shows you how to set up, compile and run a project with one instance of your Application Server. Before proceeding with these steps your server must be configured, as described in section *2– Before Starting*. Further information about configuration files and attributes is detailed section *4– Detailed Set up — Installing* .

3.1. Modules

The software is composed of several modules. The following is a short description of the components.

Table 2: List of modules

Module Name	Folder	Description
Parent	EIDAS-Parent	Module containing a consolidated and consistent location of the libraries and their version number to be used across the different modules.
Light Commons	EIDAS-Light-Commons	Light Common application component and utiliy classes used for implementing as basis for the EIDAS-Commons and MS Specific module.
Commons	EIDAS-Commons	Common Applications components and utility classes for implementing functionality of authentication service.
Encryption	EIDAS-Encryption	Encryption and signature dedicated module.
ConfigModule	EIDAS-ConfigModule	Configuration management module dedicated to facilitate eIDAS-Node configuration.
SAMLEngine	EIDAS-SAMLEngine	SAML specific Module
Specific Communication Definition	EIDAS-SpecificCommunicationDefinition	The exchange definition (interfaces) used to formalise the exchange definition between the node and the specific module.
MS specific	EIDAS-Specific	Sample of Member State (MS) specific module.
Updater	EIDAS-Updater	Module used to change configuration of a running eIDAS-Node
EidasNode	EIDAS-NODE	eIDAS-Node module (Proxy Service, Connector).
Service provide	EIDAS-SP	Sample of Service Provider module
Identity provider	EIDAS-IdP-1.0	Sample of Identity Provider module

The figure below shows the dependencies between the installed modules.

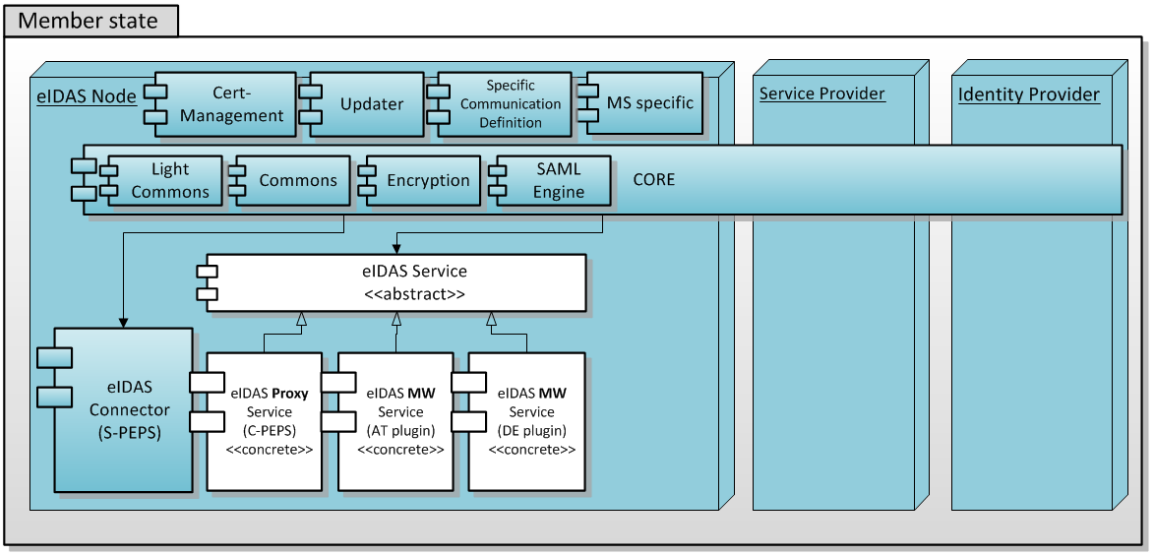


Figure 2: Dependencies between modules

The figure below shows the communications flows between modules during authentication of citizens.

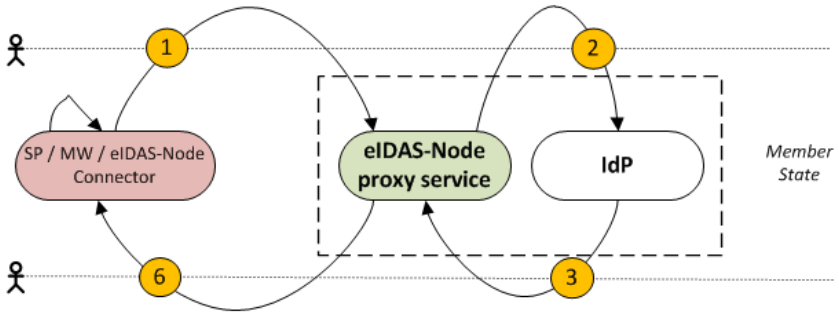


Figure 3: Communication flows between modules during authentication of citizens

3.2. Configuring the Project

To configure the project in the Basic Setup, follow the steps shown below.

3.2.1. For the eIDAS-Node

1. By default `EIDAS_CONFIG_REPOSITORY` and `SPECIFIC_CONFIG_REPOSITORY` OS evirontment or JVM command line arguments (-D option) must be set in order to specify the location of configuration files. It is possible to change or hardcode these variables in `environmentalContext.xml`.

   Note: a sample of this file is located in the EIDAS-config module.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.1.xsd">
```

```
    <!--

    Configuration repository path either from ENVIRONMENT variable or
COMMAND LINE -D option of
            EIDAS_CONFIG_REPOSITORY and
            SPECIFIC_CONFIG_REPOSITORY
         For any other option - like hard-coded values - modify this
file.
          Hard coding example:

    <bean id="eidasConfigRepository" class="java.lang.String">
        <constructor-arg value="c:/PGM/projects/configEidas/glassfish/"
/>
    </bean>

    <bean id="eidasSpecificConfigRepository" class="java.lang.String">
        <constructor-arg
value="c:/PGM/projects/configEidas/glassfish/specific/" />
    </bean>

    -->

    <bean id="eidasConfigRepository" class="java.lang.String">
        <constructor-arg value="#{
systemProperties['EIDAS_CONFIG_REPOSITORY'] ?:
systemEnvironment['EIDAS_CONFIG_REPOSITORY'] }" />
    </bean>

    <bean id="eidasSpecificConfigRepository" class="java.lang.String">
        <constructor-arg value="#{
systemProperties['SPECIFIC_CONFIG_REPOSITORY'] ?:
systemEnvironment['SPECIFIC_CONFIG_REPOSITORY'] }" />
    </bean>

</beans>
```

2. Edit the file eidas.xml to specify the following eIDAS-Node Connector and eIDAS-Node Proxy Service configuration properties:

```
connector.assertion.url=
http://insert.your.ip.here:portGoesHere/EidasNode/ColleagueResponse
connector.destination.url=
http://insert.your.ip.here:portGoesHere/EidasNode/ServiceProvider
service1.url=
http://insert.your.ip.here:portGoesHere/EidasNode/ColleagueRequest
service.specificidpredirect.url=
http://insert.your.ip.here:portGoesHere/EidasNode/IdpResponse
```

3. SP reads the configuration from file `SP_CONFIG_REPOSITORY/sp.properties` so the SP_CONFIG_REPOSITORY environment variable / command line parameter should be set up. Change the following in `sp.properties`:
   - `sp.url=http://your.ip.goes.here:portGoesHere/SP/ReturnPage`
   - `countryX.url=http://your.ip.goes.here:portGoesHere /EidasNode/ServiceProvider`
4. Edit the file eidas_Specific.xml and change the following property:
   - `idp.url= http://insert.your.ip.here:portGoesHere/IdP/AuthenticateCitizen`

You now have a Service Provider and eIDAS-Node Connector configured to run on your IP address and a eIDAS-Node Proxy Service, IdP configured to run on localhost.

4. Detailed Set up — Installing eIDAS-Nodes

There are two main types of eIDAS-Node: Connector and Proxy Service.

4.1. eIDAS-Nodes Configuration Files

This section provides a detailed description of the eIDAS-Nodes configuration files and their properties.

The `eidas.xml` file contains the properties to configure:

- General purpose parameters
- eIDAS-Node Connector
- eIDAS-Node Proxy Service

4.1.1. General purpose parameters

Table 3 lists general purpose parameters which include additional checks and security configurations.

Table 3: General purpose parameters

E=eIDAS S=STORK compliant	Key	Description
E	`metadata.activate`	Allows activation/deactivation of SAML metadata (this parameter activates/deactivates metadata publishing and requesting on both Connector and Proxy Service (see also section 9 — *eIDAS-Node SAML metadata.)*
E	`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://*server:port*/EidasNode/ConnectorResponderMetadata Will be used as Issuer in the responses that eIDAS-Node Connector sends to SP.
E	`node.metadata.not.signed.descriptors`	List of URLs corresponding to entity descriptors whose signatures have not to be checked. The format to use is http://descriptorurl1;https://descriptorurl2;...
E,S	`response.encryption.mandatory`	When set to 'true' the node will try to encrypt assertions in the generated SAML responses (provided that the encryption related configuration is in place). **Note**: this parameter is used by both Proxy Service and Connector nodes.

E=eIDAS S=STORK compliant	Key	Description
E	check.mandatory.eidas.attributes	When set to 'true' the node will check if at least one set of mandatory attributes is included in the request or in the response. **Note:** this parameter is used by both Proxy Service and Connector nodes
E	disable.check.mandatory.eidas.attributes	When set to false, the ILightRequest is checked if there are Representative attributes requested, and reject the authentication request. Default is false.
E,S	distributedMaps	When set to 'true' the node will use Hazelcast implementation for request-reply map correlations and anti-replay cache.
E,S	nonDistributedMetadata.retention	Retention period for simple metadata cache in seconds. (Note: for distributed environment it's not used, set I up in hazelcast.xml instead)

4.1.2. Attribute registry

Attribute registry holds and supplies information of types, value format and namespace for creating and validating requests and responses. The registry basically contains Attribute Definition objects built from custom XML files. There are two kind of XML files:

- one holds eIDAS protocol restricted attributes (`saml-engine-eidas-attributes.xml`); and
- the other supports additional attributes, sometimes referred as dynamic attributes (`saml-engine-additional-attributes_EidasNode.xml`).

Each Protocol Engine has its own configuration files,specified by `SamlEngine.xml` files.

The following is an example code to introduce a new attribute to the XML configuration:

```
    <entry
key="19.NameUri">http://eidas.europa.eu/attributes/natural/NewSomething</entry>
    <entry key="19.FriendlyName">NEW_SOMETHING</entry>
    <entry key="19.PersonType">NaturalPerson</entry>
    <entry key="19.Required">false</entry>
    <entry
key="19.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</e
ntry>
    <entry key="19.XmlType.LocalPart">NewSomethingType</entry>
    <entry key="19.XmlType.NamespacePrefix">eidas-natural</entry>
```

For the `key` prefix number, take the last one and increment it. For eIDAS protocol the person type (natural or legal) must be specified and aligned with namespace.

4.1.2.1. Attribute registry validation and metadata support

Besides the Attribute Registry XML files there is a hard coded list of supported core attributes in `LegalPersonSpec` and `NaturalPersonSpec` collected together in `EidasSpec` class, can be found in the saml-engine module. This is necessary to get a reference of attribute definitions to perform business rule-based validations on requests and replies.

Supported attributes are published in the Metadata.

4.1.3. eIDAS-Node Connector Configuration

The eIDAS-Node Connector configuration is composed of the following parts:

- Service Provider configuration
- eIDAS-Node Connector dedicated information
- Configuration of the recognised Connector

4.1.3.1. Service Provider Configuration

To configure the Service Provider, you must provide a value for the properties.

Table 4: eIDAS-Node Connector and SP Policies

E=eIDAS S=STORK compliant	Key	Description
S	`spId.qaalevel`	QAA level of the Id of the Service Provider e.g. `ES-SP.qaalevel=3`
E,S	`sp.default.parameters`	The default parameters for not listed Service Providers, these are the parameters they can access (`none|all|list;Of;Parameters`)
E,S	`SP-ID`	If a Service Provider is listed, these are the properties that it can access (`none|all|list;Of;Parameters`). `SP-ID` is the Service Provider ID.
E,S	`sp.authorized.parameters`	Parameters supported by this eIDAS-Node Connector (`none|all|list;Of;Parameters`)

E,S	`SP-ID.validation`	The SP's keystore alias. This must match the SP's keystore alias or use "none" to bypass this validation e.g. `ES-SP.validation=ES-SP-CERT`.
		If you need to configure the same certificate for several SPs, then just add once the Certificate and put the keystore alias in this property for each SP e.g.
		`ES-SP1.validation= ES-SP-CERT;`
		`ES-SP2.validation= ES-SP-CERT;`
		`ES-SPn.validation= ES-SP-CERT`

S	`connector.spInstitution`	Default SP Institution value to be used (only) on the Country Selector Interface.
S	`connector.spApplication`	Default SP Application value to be used (only) on the Country Selector Interface.

E=eIDAS S=STORK compliant	Key	Description
S	connector.spCountry	The eIDAS-Node Connector country ID in ISO 3166-1 alpha-3 format e.g. PT is the ISO 3166 code for Portugal.
S	connector.spSector	Default SP Sector value to be used (only) on the Country Selector Interface.
E,S	active.module.connector	Allows deactivating eIDAS-Node Connector functionality: when setting this parameter to false, eIDAS-Node Connector will answer with an error message to incoming requests. The default value is true.

4.1.3.2. eIDAS-Node Connector Dedicated Information

To identify the eIDAS-Node Connector, the following information needs to be provided.

Table 5: eIDAS-Node Connector dedicated information

E=eIDAS S=STORK compliant	Key	Description
E,S	connector.id	Id of this eIDAS-Node Connector
S	connector.assertion.url	URL of the Action to be called when returning from eIDAS-Node Proxy Service (This is the AssertionConsumerServiceURL which will be present on SAML Request and be used as destination on SAML Response)
E,S	connector.destination.url	URL of the Action to handle SP requests (This is the Service Provider URL)
E,S	saml.sp	Name of the SAML configuration to use between SP and eIDAS-Node Connector.
E,S	saml.service	Name of the SAML configuration to use between Connector and Proxy Service.
E	connector.node.url	eIDAS-Node Connector URL to be shown in site's metadata.
E	connector.contact.support.email	Email address of the support contact
E	connector.contact.support.company	Company name of the support contact
E	connector.contact.support.givenname	Given name of the support contact
E	connector.contact.support.surname	Surname of the support contact
E	connector.contact.support.phone	Phone number of the support contact
E	connector.contact.technical.email	Email address of the technical contact
E	connector.contact.technical.company	Company of the technical contact
E	connector.contact.technical.givenname	Given name of the technical contact
E	connector.contact.technical.surname	Surname of the technical contact
E	connector.contact.technical.phone	Phone number of the technical contact
E	connector.metadata.url	The URL at which the metadata of eIDAS-Node Connector will be made available, e.g. http://*server*:*port*/EidasNode/ConnectorMetadata Will be used as Issuer in the requests that eIDAS-Node Connector sends

E=eIDAS S=STORK compliant	Key	Description
E	connector.organization.name	Name of the organization displayed in Metadata
E	connector.organization.displayname	Localised display name of the organization for Metadata
E	connector.organization.url	URL of the organization for Metadata containing information
E	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://localhost:8080/EidasNode/ServiceProvider
E	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://localhost:8080/EidasNode/ServiceProvider

If you are running tests across the network you must change the connector.assertion.url and connector.destination.url to reflect the IP address of the machine running the eIDAS-Node Connector:

http://*connector.ip.address*:*connector.port.number*/*node.deployment.name*/ColleagueResponse and http://*connector.ip.address*:*connector.port.number*/*node.deployment.name*/ServiceProvider respectively.

4.1.3.3. Configuration of the Recognised eIDAS-Node Proxy Service

To add a new eIDAS-Node Proxy Service, increment the service.number and add their keys and respective values. The URL must be in the format:
http://*service.ip.address:service.port.number*/*service.deployment.name*/ColleagueRequest

Table 6: Adding eIDAS-Node Proxy Service

E=eIDAS S=STORK compliant	Key	Description
E,S	service.number	Number of known eIDAS-Node Proxy Service
E,S	serviceX.id	Id of the eIDAS-Node Proxy Service X(=unique positive integer)
E,S	serviceX.name	Name of the eIDAS-Node Proxy Service X(=unique positive integer)
E,S	serviceX.url	URL of the eIDAS-Node Proxy Service X(=unique positive integer)

E=eIDAS S=STORK compliant	Key	Description
E	serviceX.metadata.url	URL where the eIDAS-Node Proxy Service X publishes its metadata
E	serviceX.skew.notbefore	Time scew in milliseconds to adjust notBefore SAML condition in Connector. The actual value is added to the received time condition, negative value is possible.
E	serviceX.skew.notonorafter	Time scew in milliseconds to adjust notOnOrAfter SAML condition in Connector. The actual value is added to the received time condition, negative value is possible.

4.1.4. eIDAS-Node Proxy Service Configuration

When the eIDAS-Node Proxy Service plugin is active the following properties need to be provided to configure the eIDAS-Node Proxy Service component of the software.

Table 7 : Adding eIDAS-Node Proxy Service

E=eIDAS S=STORK compliant	Key	Description
E,S	service.id	Id of the eIDAS-Node Proxy Service to configure.
E,S	service.countrycode	The eIDAS-Node Proxy Service country ID in ISO 3166-1 alpha-3 format e.g. PT is the ISO 3166 code for Portugal.
S	service.maxQAAlevel	Max QAA Level that eIDAS-Node Proxy Service can authenticate (see Appendix B — eIDAS Levels of Assurance).
E,S	service.askconsent.type	Has consent been given to display consent-type form? (true/false)
E,S	service.askconsent.value	Has consent been given to display consent-value form? (true/false)
E	service.askconsent.all.attributes	Has consent been given to display all the attributes, i.e. the eIDAS attributes and the additional ones? (true/false) true: to display all the attributes false: to display only the eIDAS attributes
E	service.askconsent.attribute.names.only	Has consent to display only attribute names or name-value pairs? (true/false)
E,S	service.specificidpredirect.url	URL of the Action to be called when returning from IdP
E,S	service.citizenConsentUrl	The URL where the user provides the Citizen Consent (for information on citizen consent see Appendix C)
E,S	serviceX.skew	Skew time between the eIDAS-Node Connector and eIDAS-Node Proxy Service X in millisecond, see explanation below.
E	service.node.url	Proxy Service URL to be shown in site's metadata.

E=eIDAS S=STORK compliant	Key	Description
E	service.contact.support.email	Email address of the support contact
E	service.contact.support.company	Company of the support contact
E	service.contact.support.givenname	Given name of the support contact
E	service.contact.support.surname	Surname of the support contact
E	service.contact.support.phone	Phone number of the support contact
E	service.contact.technical.email	Email address of the technical contact
E	service.contact.technical.company	Company name of the technical contact
E	service.contact.technical.givenname	Given name of the technical contact
E	service.contact.technical.surname	Surname of the technical contact
E	service.contact.technical.phone	Phone number of the technical contact
E	service.organization.name	Name of the orgvanisation displayed in the metadata
E	service.organization.displayname	Localised display name of the organisation for Metadata
E	service.organization.url	URL of the organisation for Metadata containing information
E	service.metadata.url	The URL under which the metadata of Proxy Service will be made available, e.g. http://*server:port*/EidasNode/ServiceMetadata Will be used as Issuer in the requests that Proxy Service sends
E	service.LoA	Sets the Level of Assurance for the service. The following values are accepted: http://eidas.europa.eu/LoA/low http://eidas.europa.eu/LoA/substantial http://eidas.europa.eu/LoA/high
E	service.requester.metadata.url	The URL where the metadata of Proxy Service (presenting itself as an SP) will be made available, e.g. http://EidasNode:8888/EidasNode/ServiceRequesterMetadata . It will be used as Issuer in the requests that eIDAS-Node Connector sends to IdP.

E=eIDAS S=STORK compliant	Key	Description
E	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-POST. e.g. http://EidasNode:8888/EidasNode/ColleagueRequest
E	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-Redirect. e.g. http://EidasNode:8888/EidasNode/ColleagueRequest

If you are running tests across the network you must change the:

service.specificidpredirect.url

to reflect the IP address of the machine running **eIDAS-Node Proxy Service**, respectively the URL formats must be:

http://*service.ip.address*:*service.port.number* /*service.deployment.name*/IdpResponse

4.1.4.1. eIDAS-Node Proxy Service activation/deactivation

Table 8: Adding a plugin

Key	Description
active.module.service	Whether to activate the Proxy Service module or not. Possible values: true, false. Default value: true.

4.1.4.2. Additional Configuration — Skew Time

It is possible for clocks to be out of synchronisation between eIDAS-Node instances (Proxy Service / Connector). To prevent validation errors occurring in the Connector you can configure a skew time for each Proxy Service. The skew time gives the Connector an additional tolerance window for validating the timestamps in the SAML Responses that are sent by the Proxy Service.

Table 9: Security Policies

E=eIDAS S=STORK compliant	Key	Description
E,S	max.requests.ip	Maximum limit of requests per IP within the time frame of max.time.ip (-1 unlimited)

E=eIDAS S=STORK compliant	Key	Description
E,S	max.requests.sp	Maximum limit of requests per SP within the time frame of max.time.sp (-1 unlimited)
E,S	max.time.ip	Time frame for IP requests (seconds)
E,S	max.time.sp	Time frame for SP requests (seconds)
E,S	trusted.sp.domain	Allowed SPs to communicate with the eIDAS-Node Connector (none|all|list;Of;Domains)
E,S	validation.bypass	Bypass all SP validations (true|false)
E,S	validation.method	Validate the Service Provider by domain or by domain and spId (domain|spid)
S	min.qaaLevel.value	Minimum valid QAA level.
S	max.qaaLevel.value	Maximum valid QAA level.

4.1.4.3. Additional Configuration — Security

Table 10: Security HTTP header parameters

E=eIDAS S=STORK compliant	Key	Description
E,S	security.header.CSP.enabled	Enable/disable sending the Content Security Policy (CSP) header. CSP protects against the injection of foreign content (refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features).
E,S	security.header.CSP. includeMozillaDirectives	In the CSP, this additional directive can be added for backward compatibility with old Mozilla browsers (refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features).
E,S	security.header. XXssProtection.block	This header enables the cross-site-scripting (XSS) filter built into most recent web browsers (refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features).
E,S	security.header. XContentTypeOptions.noSniff	The only defined value 'nosniff' prevents Internet Explorer and Google Chrome from 'MIME-sniffing' by inspecting the content of a response (refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features).

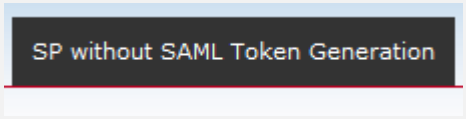
E=eIDAS S=STORK compliant	Key	Description
E,S	`security.header. XFrameOptions.sameOrigin`	Prevents the application from being propagated in a frame or iframe, which in turns protects against key logging, clickjacking and similar attacks. Setting this option to **true** will prevent the eIDAS-Node from being framed in another application. If the SP needs to frame the eIDAS-Node, the option has to be set to 'false' (such as on the second tab of the SP Demo where the SAML request is generated by the eIDAS-Node).  Refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features.
E,S	`security.header. HSTS.includeSubDomains`	HTTP Strict-Transport-Security (HSTS) instructs browsers to prefer secure connections to the server (HTTP over SSL/TLS) over insecure ones (refer to section 13 — *eIDAS-Node Security Headers* for more information about the security features).
E,S	`security.header.CSP.fallback CheckMode`	If enabled, CSP fallback check mode includes an enforced CSP violation in JSP pages in order to check browser CSP feature. The included script displays a warning message in client browsers if CSP is not supported. However with CSP enabled browsers it may result in flood of warning messages logged by CSP report servlet. Disabled by default.

Table 11: Check on certificate security parameter

E=eIDAS S=STORK compliant	Key	Description
E,S	`check.citizenCertificate. serviceCertificate`	Checks that the country code stored in the eIDAS-Node Proxy Service SAML signing certificate is the same as the citizen country code in the SAML authentication request.

Table 12: Configuring encryption algorithm

E=eIDAS S=STORK compliant	Key	Description

E,S	`data.encryption.algorithm`	Contains the encryption algorithm to be used by Proxy Service and Connector. Possible value must be :
		
		`<entry key="data.encryption.algorithm"></entry>`
		`<!-- List of Encryption algorithms`
		
		http://www.w3.org/2009/xmlenc11#aes128-gcm;
		http://www.w3.org/2009/xmlenc11#aes256-gcm;
		http://www.w3.org/2009/xmlenc11#aes192-gcm;

E,S	`encryption.algorithm.whitelist`	Contains the encryption algorithms allowed in the responses received by eIDAS-Node components. As per specification, this should be: http://www.w3.org/2009/xmlenc11#aes128-gcm; http://www.w3.org/2009/xmlenc11#aes256-gcm; http://www.w3.org/2009/xmlenc11#aes192-gcm;
E,S	`check_certificate_validity_period`	Boolean value (true|false), which indicates if the application will disallow the use of obsolete certificates.
E,S	`disallow_self_signed_certificate`	Boolean value (true|false), which indicates if the application will disallow of the use of self-signed certificates.
E,S	`response.encryption.mandatory`	Boolean value (true/false), which indicates if the application will force the encryption of the SAML Response.

4.1.4.4. Additional Configuration — SignModule_Service.xml and SignModule_Connector.xml

It may be necessary to change the `keyStorePath` to reflect the location of your `eidasKeyStore.jks` file, please see section *10 — eIDAS-Node SAML Engine* for more information.

4.1.4.5. Additional Configuration — Anti-replay Cache and Correlation Map Configuration

To prevent a replay of SAML requests an anti-replay cache is implemented at the eIDAS-Node Connector and eIDAS-Node Proxy Service level. We provide two different implementations for these caches, which can be configured. By default, the eIDAS-Node is set up to use a distributed cache with expiration.

This implementation is provided for correlating request and reply pairs both for `AuthenticationRequest`s and `LightRequest`s.

Hazelcast backed caches are intended to be used in production environments. Development environment may use lighter cache implementations (simple

ConcurrentHashMap based), which are activated by setting the parameter distributedMaps to "false" (without quotes), in eidas.xml.

By default there is one Hazelcast instance used by the node for both correlation and anti-replay map purposes.

```
<!-- production environment hazelcast instance name -->
<bean id="defaultHazelcastInstance" class="java.lang.String">
   <constructor-arg value="eidasHazelcastInstance"/>
</bean>
```

Figure 4: Default Hazelcast instance name

The default instance is provided by the eidasHazelcastInstanceInitializer bean.

```
<!-- production environment hazelcast initializer bean - injected into map
providers -->
<bean id="eidasHazelcastInstanceInitializer" class="
eu.eidas.auth.commons.cache.HazelcastInstanceInitializer" init-
method="initializeInstance" lazy-init="true">
   <property name="hazelcastConfigfileName" value="hazelcast.xml"/>
   <property name="hazelcastInstanceName" ref="defaultHazelcastInstance"/>
</bean>
```

Figure 5: Default Hazelcast instance provider been

This bean is to be injected into ConcurrentMapServiceDistributedImpl and ConcurrentMapServiceDistributedImpl beans. If the distributed environment requires setup of multiple Hazelcast instances, the configuration can be done simply adding more of the above beans to applicationContext and specificApplicationContext files.

```
<bean id="springServiceCMapAntiReplayProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl"
lazy-init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName" value="antiReplayCacheService"/>
</bean>
<bean id="springConnectorCMapAntiReplayProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl"
lazy-init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
   <property name="cacheName" value="antiReplayCacheConnector"/>
</bean>
```

Figure 6: Anti-replay cache configuration — Hazelcast — applicationContext.xml

For correlation maps, there are three AuthRequest type maps in ApplicationContext, one for the Connector, two for the Proxy Service one of which is for the Specific.

```
<bean id="springServiceCMapspecificSpCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
```

```
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName" value="specificSpRequestCorrelationCacheService"/>
</bean>
<bean id="springConnectorCMapCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName" value="connectorRequestCorrelationCacheService"/>
</bean>
<bean id="springServiceCMapCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName"
value="proxyServiceRequestCorrelationCacheService"/>
</bean>
```

Figure 7: Correlation map cache configuration — Hazelcast — applicationContext.xml

For the Specific part, `specificSpRequestCorrelationMap`, the map instance must be the same as used in the eIDAS-Node (`springServiceCMapspecificSpCorProvider`). `LightRequest` map types are defined here.

```
<bean id="springServiceCMapspecificIdpCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName" value="specificIdpRequestCorrelationCacheService"/>
</bean>
<!-- LightRq correlation maps -->
<bean id="springConnectorCMapspecificLightCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName"
value="specificConnectorLtRequestCorrelationCacheService"/>
</bean>
<bean id="springServiceCMapspecificLightCorProviderProd"
class="eu.eidas.auth.commons.cache.ConcurrentMapServiceDistributedImpl" lazy-
init="true">
  <property name="hazelcastInstanceInitializer"
ref="eidasHazelcastInstanceInitializer"/>
  <property name="cacheName"
value="specificServiceLtRequestCorrelationCacheService"/>
</bean>
```

Figure 8: Correlation map cache configuration — Hazelcast — specificApplicationContext.xml

For more information about the Hazelcraft product, please refer to Appendix D.

4.1.4.6. Signature

E=eIDAS S=STORK compliant	Key	Description
E,S	`signature.algorithm`	The signing algorithm (SHA2 based) used by the default signer for outgoing requests. Possible values: http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 http://www.w3.org/2001/04/xmldsig-more#rsa-sha384 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 The default value is: http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 If another value is set, eIDAS-Nodes will use RSA-SHA512 algorithm and an error will be logged.
E,S	`signature.algorithm.whitelist`	The list of allowed signature algorithms (in incoming requests). It contains OpenSAML's supported signing algorithms, separated by **;**.Currently the elements of the list s may be picked from the following: http://www.w3.org/2001/04/xmldsig-more#rsa-sha256 http://www.w3.org/2001/04/xmldsig-more#rsa-sha384 http://www.w3.org/2001/04/xmldsig-more#rsa-sha512 http://www.w3.org/2001/04/xmldsig-more#rsa-ripemd160 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha256 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha384 http://www.w3.org/2001/04/xmldsig-more#ecdsa-sha512
E,S	`Response.sign.assertions`	When set to true, the SAML Responses (generated in Proxy Service and Connector) will have the attribute assertion signed

4.1.4.7. SAML Binding method

E=eIDAS S=STORK compliant	Key	Description
S	`allow.redirect.binding`	Whether to allow the HTTP Redirect binding. Possible values are true/false.(this was only applicable for STORK 1 message format and for testing purposes). For eIDAS, there are no bindings in the request.
E,S	`validate.binding`	Whether to validate the actual binding (POST or GET/Redirect) against `ProtocolBinding` attribute value of the SAML request. Possible values are true/false.

By default, eIDAS-Nodes operate using SAML Post Binding. The parameter `allow.redirect.binding` (set to true) instructs the eIDAS-Node to accept HTTP Redirect Binding SAML requests, normally coming as HTTP GET requests. When HTTP Redirect Binding is used the following items should be considered:

- Most browsers have low limit for the size of GET request.
- Most servers have low limit for the size for HTTP header (e.g. in Apache Tomcat v7 this limit is about 8k; in order to increase this limit, the connector

element in server.xml should contain a maxHttpHeaderSize element with the new limit);

- When this binding is activated, an HTTP redirect binding request received by Connector will be forwarded also as a redirect to Proxy Service and further (to IdP);
- The response is always sent back through a HTTP Post operation.

4.1.4.8. Error Codes and Error Messages

The full list of eIDAS-Node error codes and related error messages is shown in Appendix *A*. Each error message must be used to match the error to present to the citizen (`errors.properties` file), to present to `sysadmin` (`sysadmin.properties`) and to translate in the Connector the errors from the Proxy Service.

For each error message a new property should exist in the following files:

- `EIDAS-NODE\src\main\resources\error.properties`
- `EIDAS-NODE\src\main\resources\sysadmin.properties`
- `EIDAS-NODE\src\main\resources\eidastranslation.properties`

For example, for the following `eidasErrors.properties` property:

```
connectorSAMLResponse.message=error.gen.connector.saml
```

you must add the following in the `error.properties`:

```
authenticationFailed.code=003002
authenticationFailed.message=authentication.failed
```

You must also add the following property to `sysadmin.properties` in the native Proxy Service language:

```
authentication.failed={0} - Authentication Failed.
```

Note: This format is mandatory: `{0} - Error Message`.

Using the same format, you must add the following property to `eidastranslation.properties` in the native eIDAS-Node Connector language:

```
authentication.failed={0} - A autenticação falhou.
```

Bear in mind that you must have as many error.properties files as the required languages. The file name follows the standards:

- `error_pt.properties` (i.e. Portuguese language)
- `error_es.properties` (i.e. Spanish language)
- `error_en.properties` (i.e. English language)

4.1.5. Specific Properties

The `eidas_Specific.xml` is the configuration file that holds all the MS-Specific configurations. The location of this file must be set by the `SPECIFIC_CONFIG_REPOSITORY` environment variable or command line argument.

Table 13: Providers' Specific Properties

E=eIDAS S=STORK compliant	Key	Description
E,S	external.authentication	Indicates if the authentication is external
E,S	idp.url	URL of the Identity Provider
S	derive.isAgeOver.impl	Class that implements isAgeOver derivation.
S	derive.age.impl	Class that implements age derivation.
E,S	derive.eid.impl	Class that implements eId derivation.

A Specific demonstration is provided by way of Identity Provider:

- You must configure the IP address of the machine running the Identity Provider using the format: http://***idp.ip.address***:*idp.port.number*/***idp.deployment.name***/Authenticate Citizen

4.1.5.1. Derivation

Table 14: Derivation

E=eIDAS S=STORK compliant	Key	Description
E,S	deriveAttr.number	Number of derivation attributes
E,S	deriveAttrX.id	Id of the derivation attribute X(= positive integer)
E,S	deriveAttrX.name	Name of the derivation attribute X(= positive integer)

4.1.5.2. Permitted Attributes Values

Table 15: Permitted attributes values

E=eIDAS S=STORK compliant	Key	Description
E,S	attrValue.number	Number of attributes' values to validate.
E,S	attrValue.id	Id of the attribute X(= positive integer)
E,S	attrValue.name	Name of the attribute X(= positive integer)
E,S	attrValue.value	The allowed attribute's values (empty if attribute value is checked on the implementation class).

Derivation attributes, are attributes that cannot be sent directly to the Identity Provider, but must be translated to other attributes before being sent.

Table 16: Derivation attributes

E=eIDAS S=STORK compliant	Key	Description
E,S	allow.unknowns	If set to true will allow unknown values to be normalized.
E,S	allow.derivation.all	If set to true will allow the derivation of derived attributes, as opposed to, the derivation of the main attribute.

Table 17: Examples of derived attributes

E=eIDAS S=STORK compliant	Key	Description
S	dateOfBirth.stork.format	Defines the date format from the AP/IdP, currently 'yyyyMMdd'.
S	dateOfBirth.hasSeparator	Does the specific date has a separator char?
S	dateOfBirth.specific.separator	Defines the specific separator for the date.
S	dateOfBirth.implemention	Class that implements the dateOfBirth value normalization.

4.1.5.3. Attribute Value Validation

Some attribute values must be validated before the Proxy Service creates the response to the eIDAS-Node Connector. The following table shows the classes for the attributes that must be validated.

Table 18: Sample of specific validation parameter

E=eIDAS S=STORK compliant	Key	Description
S	validate.gender.impl	Class that implements the gender attribute value validation.
S	validate.countryCodeOfBirth.impl	Class that implements the countryCodeOfBirth attribute value validation.
S	validate.nationalityCode.impl	Class that implements the nationalityCode attribute value validation.
S	validate.maritalStatus.impl	Class that implements the maritalStatus attribute value validation.

4.1.6. Service Provider

The Service Provider `sp.properties` configuration details are described in the following table. The location of this file must be set by the `SP_CONFIG_REPOSITORY` environment variable or command line argument

Table 19: Service Provider Properties

E=eIDAS S=STORK compliant	Key	Description
E,S	`provider.name`	Name for this Service Provider
S	`sp.sector`	Sector for this Service Provider
S	`sp.application`	Application for this Service Provider
E,S	`sp.return`	URL used when the eIDAS-Node Connector finishes the process. This must be the value of the machine running the Service Provider, its format is http://*sp.ip.address*:*sp.port.number*/*sp.deployment.name*/ReturnPage .
E,S	`sp.qaalevel`	QAA level of this Service Provider
E,S	`sp.country`	Country for this Service Provider
E	`sp.metadata.url`	The URL used to provide metadata (i.e. https://sp:8888/SP/metadata)
E	`sp.metadata.httpfetch`	Enables fetching of metadata from Specific-Connector. Enabled by default, if disabled, SP.metadata.repository.path must be set to a local metadata file to be processed.
E	`sp.metadata.repository.path`	If httpfetch is disabled, SP will read and monitor the specified direcotry for metadata files (any XML) instead of checking Specific-Connector provided metadata.
E	`sp.metadata.retention`	Retention period for SP internal Metadata cache, seconds.
E	`sp.metadata.validatesignature`	Check fetched or loaded metadata signature, before cached and used locally. Entity descriptors for URLs specified in sp.metadata.trusteddescriptors are not verified. Default: true.
E	`sp.metadata.trusteddescriptors`	List of entity descriptor URLs. Metadata from these sources are not being verified by signature checking.

It is also possible to specify SP-provided metadata content in this file by setting:

- `contact.support.email, contact.support.company;`
- `contact.support.givenname, contact.support.surname;`
- `contact.support.phone, contact.technical.email;`
- `contact.technical.company;`

- `contact.technical.givenname, contact.technical.surname;`
- `contact.technical.phone;`
- `organization.name;`
- `organization.displayname;`
- `organization.url;`
- `signature.algorithm.whitelist`; and
- `encryption.algorithm.whitelist`.

The following table describes the available eIDAS-Node for this Service Provider.

Table 20: Available eIDAS-Node for Service Provider

E=eIDAS S=STORK compliant	Key	Description
E,S	`country.number`	The number of possible eIDAS-Nodes that can communicate with this SP
E,S	`countryX.name`	The name of the eIDAS-Node X(= positive integer)
E,S	`countryX.url`	The URL for the eIDAS-Node X. This must be the value of the machine running the eIDAS-Node using the format: http://*node.ip.address:node.port.number/node.deployment.name*/.
E,S	`countryX.countrySelector`	The URL for the `CountrySelector` of the eIDAS-Node X. This must have the value of the machine running the Service Provider, its format is http://*node.ip.address:node.port.number /sp.deployment.name*/CountrySelector

4.1.7. Identity Provider

The `user.properties` holds the credentials for citizens who are able to log in. The format is: `<username>=<password>`.

If the citizen does not have an AP, the IdP can be used as a replacement. The `idp.properties` is used by the IdP to provide the attribute values in the format: `<username>.<attributeName>=<attributeValue>`.

Table 21: Sample of user.properties content

Key	Description
`myUser=myPassword`	A sample username and password
`myUser.LegalName=my legal name`	A sample attribute definition

The `idp.properties` holds configuration parameters about the application, especially metadata configuration. The location of this file must be set by the `IDP_CONFIG_REPOSITORY` environment variable or command line argument

Table 22: Identity Provider Properties

Key	Description
idp.metadata.url	The URL used to provide metadata (i.e. https://<idp.*yourHostname*>:<idp.*yourPort*>/IdP/metadata)
idp.metadata.httpfetch	Enables fetching of metadata from Specific-Proxy. Enabled by default, if disabled, idp.metadata.repository.path must be set to a local metadata file to be processed.
idp.metadata.repository.path	If httpfetch is disabled, IDP will read and monitor the specified direcotry for metadata files (any XML) instead of checking Specific-Proxy provided metadata.
idp.metadata.retention	Retention period for IdP internal Metadata cache, seconds.
idp.metadata.validatesignature	Check fetched or loaded metadata signature, before cached and used locally. Entity descriptors for URLs specified in sp.metadata.trusteddescriptors are not verified. Default: true.
idp.metadata.trusteddescriptors	List of entity descriptor URLs. Metadata from these sources are not being verified by signature checking.
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. https://idp:8080/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. https://idp:8080/IdP/AuthenticateCitizen

As with the SP, the following IdP metadata content can be changed in the file idp.properties:

- contact.support.email;
- contact.support.company;
- contact.support.givenname;
- contact.support.surname;
- contact.support.phone;
- contact.technical.email;
- contact.technical.company;
- contact.technical.givenname;
- contact.technical.surname;
- contact.technical.phone;
- organization.name;
- organization.displayname;

- `organization.url`;
- `signature.algorithm.whitelist`; and
- `encryption.algorithm.whitelist`.

4.1.8. Tomcat/GlassFish Server Deployment

You must compile, install and deploy the projects, either by compiling the aggregator project or by compiling each module separately in the order shown below. At a command prompt, navigate to the folder shown below and enter the corresponding command line:

Table 23: Aggregator Project Build for Tomcat/GlassFish Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	`mvn clean install -P tomcat`
2	EIDAS-Node	Tomcat: `copy target\EidasNode.war $TOMCAT_HOME\webapps\EidasNode.war` GlassFish: `copy target\EidasNode.war $GLASSFISH_DOMAIN\autodeploy\EidasNode.war`
3	EIDAS-SP	Tomcat: `copy target\SP.war $TOMCAT_HOME\webapps\SP.war` GlassFish: `copy target\SP.war $GLASSFISH_DOMAIN\autodeploy\SP.war`
4	EIDAS-IdP-1.0	Tomcat: `copy target\IdP.war $TOMCAT_HOME\webapps\IdP.war` GlassFish: `copy target\IdP.war $GLASSFISH_DOMAIN\autodeploy\IdP.war`

Table 24: Module Based Build for Tomcat/GlassFish Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	`mvn clean install`
2	EIDAS-Light-Commons	`mvn clean install`
3	EIDAS-Commons	`mvn clean install`
4	EIDAS-SpecificCommunicationDefinition	`mvn clean install`
5	EIDAS-Encryption	`mvn clean install`
6	EIDAS-ConfigModule	`mvn clean install`
7	EIDAS-SAMLEngine	`mvn clean install`
8	EIDAS-Updater	`mvn clean install`

Step	Folder	Command line
9	EIDAS-Specific	mvn clean install
10	EIDAS-Node	a. mvn clean package –P tomcat b. Tomcat: copy target\EidasNode.war $TOMCAT_HOME\webapps\EidasNode.war GlassFish: copy target\EidasNode.war $GLASSFISH_DOMAIN\autodeploy\EidasNode.war
11	EIDAS-SP	a. mvn clean package –P tomcat b. Tomcat: copy target\SP.war $TOMCAT_HOME\webapps\SP.war GlassFish: copy target\SP.war $GLASSFISH_DOMAIN\autodeploy\SP.war
12	EIDAS-IdP-1.0	a. mvn clean package –P tomcat b. Tomcat: copy target\IdP.war $TOMCAT_HOME\webapps\IdP.war GlassFish: copy target\IdP.war $GLASSFISH_DOMAIN\autodeploy\IdP.war

4.1.9. JBoss 6 Server Deployment

You must compile, install and deploy the projects, either by compiling the aggregator project or by compiling each module separately in the order shown below. At a command prompt, navigate to the folder shown below and enter the corresponding command line:

Table 25: Aggregator Project Build for JBoss 6 Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P jBoss6
2	EIDAS-Node	copy target\EidasNode.war $JBOSS_HOME\server\ $SERVER_CONFIG\deploy\EidasNode.war
3	EIDAS-SP	copy target\SP.war $JBOSS_HOME\server\ $SERVER_CONFIG\deploy\SP.war
4	EIDAS-IdP-1.0	copy target\IdP.war $JBOSS_HOME\server\ $SERVER_CONFIG\deploy\IdP.war

Table 26: Module Based Build for JBoss 6 Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install
2	EIDAS-Light-Commons	mvn clean install
3	EIDAS-Commons	mvn clean install
4	EIDAS-SpecificCommunicationDefinition	mvn clean install
5	EIDAS-Encryption	mvn clean install
6	EIDAS-ConfigModule	mvn clean install
7	EIDAS-SAMLEngine	mvn clean install
8	EIDAS-Updater	mvn clean install
9	EIDAS-Specific	mvn clean install
10	EIDAS-Node	a. mvn clean package -P jBoss6 b. copy target\EidasNode.war $JBOSS_HOME\server\$SERVER_CONFIG\deploy\EidasNode.war
11	EIDAS-SP	a. mvn clean package -P jBoss6 b. copy target\SP.war $JBOSS_HOME\server\$SERVER_CONFIG\deploy\SP.war
12	EIDAS-IdP-1.0	a. mvn clean package -P jBoss6 b. copy target\IdP.war $JBOSS_HOME\server\$SERVER_CONFIG\deploy\IdP.war

4.1.10. JBoss7 Server Deployment

You must compile, install and deploy the projects, either by compiling the aggregator project or by compiling each module separately in the order shown below. At a command prompt, navigate to the folder shown below and enter the corresponding command line:

Table 27: Aggregator Project Build for JBoss7 Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install -P jBoss7
2	EIDAS-Node	copy target\EidasNode.war $JBOSS_HOME\standalone\deployments\EidasNode.war

Step	Folder	Command line
3	EIDAS-SP	copy target\SP.war $JBOSS_HOME\ standalone\deployments\SP.war
4	EIDAS-IdP-1.0	copy target\IdP.war $JBOSS_HOME\ standalone\deployments\IdP.war

Table 28: Module Based Build for JBoss7 Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install
2	EIDAS-Light-Commons	mvn clean install
3	EIDAS-Commons	mvn clean install
4	EIDAS-SpecificCommunicationDefinition	mvn clean install
5	EIDAS-Encryption	mvn clean install
6	EIDAS-ConfigModule	mvn clean install
7	EIDAS-SAMLEngine	mvn clean install
8	EIDAS-Updater	mvn clean install
9	EIDAS-Specific	mvn clean install
10	EIDAS-Node	a. mvn clean package –P jBoss7P b. copy target\EidasNode.war $JBOSS_HOME\ standalone\deployments\EidasNode.war
11	EIDAS-SP	a. mvn clean package –P jBoss7 b. copy target\SP.war $JBOSS_HOME\ standalone\deployments\SP.war
12	EIDAS-IdP-1.0	a. mvn clean package -P jBoss7 b. copy target\IdP.war $JBOSS_HOME\ standalone\deployments\IdP.war

4.1.11. WebLogic Server Deployment

You must compile, install and deploy the projects, either by compiling the aggregator project or by compiling each module separately in the order shown below. At a command prompt, navigate to the folder shown below and enter the corresponding command line:

Table 29: Aggregator Project Build WebLogic Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install –P weblogic
2	EIDAS-Node	copy target\EidasNode.war $WLS_HOME\DOMAIN\ autodeploy\EidasNode.war
3	EIDAS-SP	copy target\SP.war $WLS_HOME\DOMAIN\ autodeploy\SP.war
4	EIDAS-IdP-1.0	copy target\IdP.war $WLS_HOME\DOMAIN\ autodeploy\IdP.war

Table 30: Module Based Build for WebLogic Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install
2	EIDAS-Light-Commons	mvn clean install
3	EIDAS-Commons	mvn clean install
4	EIDAS-SpecificCommunicationDefinition	mvn clean install
5	EIDAS-Encryption	mvn clean install
6	EIDAS-ConfigModule	mvn clean install
7	EIDAS-SAMLEngine	mvn clean install
8	EIDAS-Updater	mvn clean install
9	EIDAS-Specific	mvn clean install
10	EIDAS-Node	a. mvn clean package –P weblogic b. copy target\EidasNode.war $WLS_HOME\DOMAIN\ autodeploy\EidasNode.war
11	EIDAS-SP	a. mvn clean package –P weblogic b. copy target\SP.war $WLS_HOME\DOMAIN\ autodeploy\SP.war
12	EIDAS-IdP-1.0	a. mvn clean package -P weblogic b. copy target\IdP.war $WLS_HOME\DOMAIN\ autodeploy\IdP.war

4.1.12. WebSphere Server Deployment

You must compile, install and deploy the projects, either by compiling the aggregator project or by compiling each module separately in the order shown below using WebSphere's Admin Console. At a command prompt, navigate to the folder shown below and enter the corresponding command line:

Table 31: Aggregator Project Build WebSphere Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install -P websphere

Table 32: Module Based Build for WebSphere Server Deployment

Step	Folder	Command line
1	EIDAS-Parent	mvn clean install
2	EIDAS-Light-Commons	mvn clean install
3	EIDAS-Commons	mvn clean install
4	EIDAS-SpecificCommunicationDefinition	mvn clean install
5	EIDAS-Encryption	mvn clean install
6	EIDAS-ConfigModule	mvn clean install
7	EIDAS-SAMLEngine	mvn clean install
8	EIDAS-Updater	mvn clean install
9	EIDAS-Specific	mvn clean install
10	EIDAS-Node	mvn clean package -P websphere
11	EIDAS-SP	mvn clean package -P websphere
12	EIDAS-IdP-1.0	mvn clean package -P websphere

4.2. Start the Application

After performing the steps above, start your Application Server. Open your browser and navigate to http://localhost:8888/SP (if your server is listening on another port you can change the URL accordingly).

The following page opens.

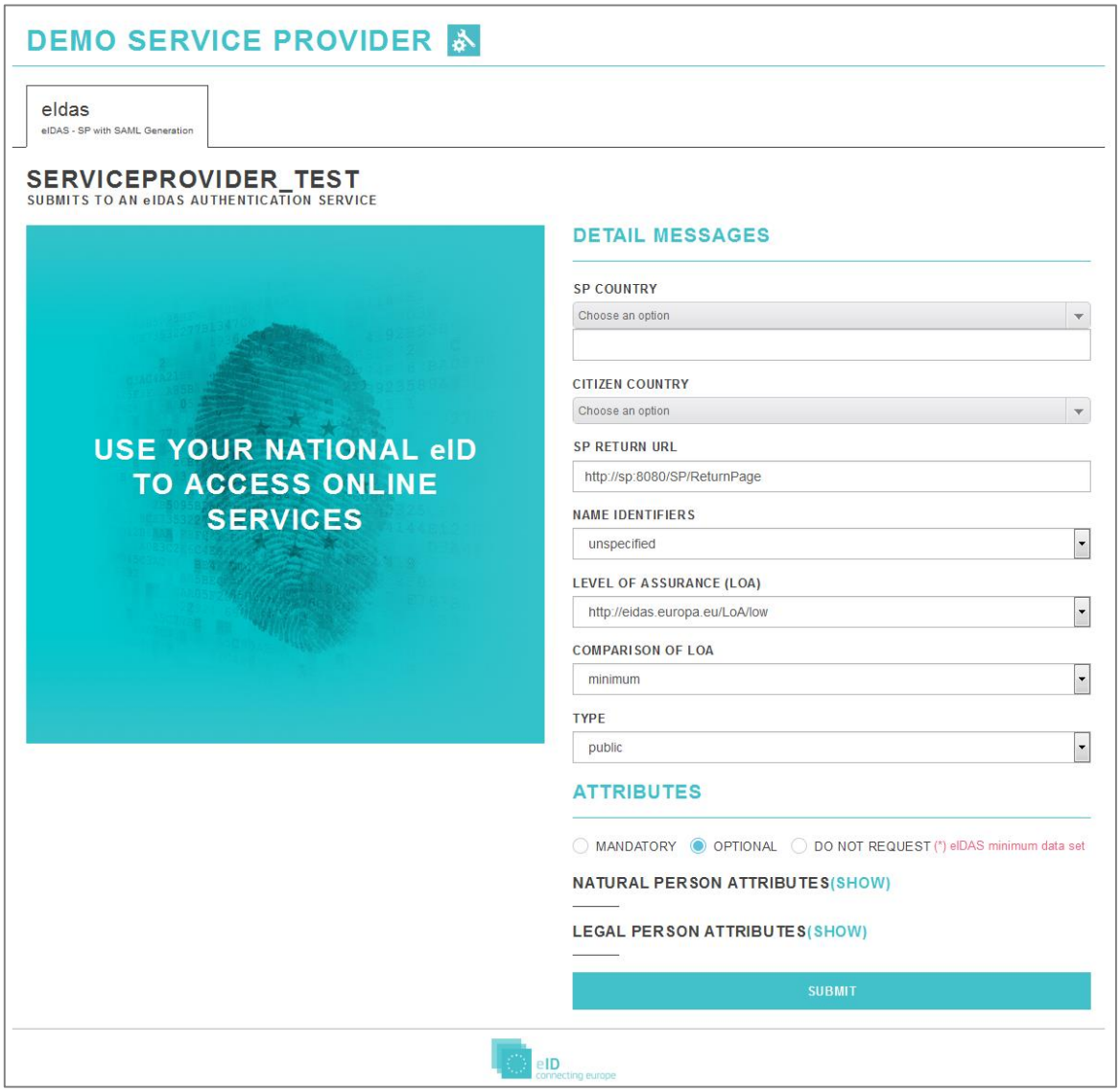


Figure 9: Service Provider Home Page

Warning: If you change the deployment names of eIDAS-Nodes, SP and IdP, you must also change the eIDAS-Node configuration as described in section *4 — Detailed Set up — Installing* to reflect those changes.

5. Server Final File Structure

This section shows the final structure of your application server relevant directories, so that you can confirm that you have made the proper configurations. The structure of the application's 'war' files are also shown so you can verify that your applications were built successfully.

5.1. Tomcat 7, 8

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

$TOMCAT_HOME\webapps\
      IdP.war
      EidasNode.war
      SP.war
      (server specific directories were not included)
```

5.2. JBoss 6

```
$JBOSS_HOME\ endorsed
      jboss-annotations-api_1.1_spec.jar
      jboss-jaxb-api_2.2_spec.jar
      jboss-jaxws-api_2.2_spec.jar
      jbossws-cxf-factories.jar
      stax-api.jar
      serializer-2.7.2.jar
      xalan-2.7.2.jar
      xercesImpl-2.11.0.jar
      xml-apis-1.4.01.jar
      xml-resolver-1.2.jar

$JBOSS_HOME\server\$SERVER_CONFIG\deploy
      IdP.war
      EidasNode.war
      SP.war
      (server specific directories were not included)
```

5.3. JBoss 7

- Check modules directory for the presence of `BouncyCastle` and `xml-apis` modules.
- Copy war files under $JBOSS_HOME\standalone\Deployments.

5.4. GlassFish V3, V4

5.4.1. GlassFish V3

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

$GLASSFISH_HOME\glassfish\domains\domain1\autodeploy\
      IdP.war
      EidasNode.war
      SP.war
      (server specific directories were not included)
```

5.4.2. GlassFish V4

```
$GLASSFISH_DOMAIN\lib\ext\
      xml-apis-1.4.01.jar

$GLASSFISH_DOMAIN\autodeploy\
      IdP.war
      EidasNode.war
      SP.war
      (server specific directories were not included)
```

5.5. WebLogic

```
$WLS_HOME\domain\autodeploy\
      IdP.war
      EidasNode.war
      SP.war
      (server specific directories were not included)

$DOMAIN_HOME\lib\
      xml-apis-1.4.01.jar
```

5.6. WebSphere Application Server

WebSphere Application Server 8.5.5 has no requirement to add/replace endorsed libraries. The deployment of the WAR files may be done using the admin console.

Note: for WebSphere Liberty Profile deployment see section 2.2.9 — *Configuring WebSphere Liberty Profile*.

6. Developing Specific Parts

This section provides information for the development of each country's specific parts.

You should begin by getting the project EIDAS-Specific, and start your development from there.

There are several classes you should consider:

- `CitizenAuthenticationBean` – The class that decides if the authentication will be performed internally or externally, as described in the following section
- `SpecificEidasService and SpecificEidasConnector` – These classes contain the details of the Member State's specific implementation.
- `SpecificProxyServiceImpl` – The class that is responsible for converting the request coming from the Proxy Service into the IdP specific protocol and. the external IdP response into the common format.
- `SpecificConnectorImpl`– The class that is responsible for converting the SP request into the common format and the Connector response into the SP response specific protocol.

6.1. Authentication

There are two authentication mechanisms, internal and external:

- Internal authentication (Internal IdP) — the request handling takes place directly on the eIDAS-Node Proxy Service.
- External authentication — the specific implementation redirects the request to an External IdP, which handles it, and then replies back to the `IdPResponseServlet` where the response will be converted to the appropriate format.

6.1.1. Implementing Internal IdP Authentication

You can implement the authentication mechanism directly in `SpecificProxyServiceImpl`.

1. Edit the file:
   eidas_Specific.xml
   in the configuration and change the property: external.authentication=**false**.
2. You should also implement the authentication logic in the function:
   sendRequest (…) on the class SpecificProxyServiceImpl, which has the following parameters:

Table 33: Internal IdP Authentication parameters

Parameters	Description
`ILightRequest lightRequest`	The data transfer object exchanged between the eIDAS protocol and the specific protocol
`HttpServletRequest httpServletRequest`	The HTTP Servlet Request
`HttpServletResponse httpServletResponse`	The HTTP Servlet Response

6.1.2. Implementing External IdP Authentication

To specify that authentication is to be handled by an external IdP:

1. Edit the file:
   eidas_Specific.xml
   and change the property: external.authentication=**true**.
2. Change the property `idp.url` to the URL that will handle the authentication, (e.g. `idp.url=http://idp/IdP/AuthenticateCitizen`).

6.2. Getting Attributes

The requested attributes can be obtained from the IdP — the attributes may be provided by the IdP.

7. eIDAS-Node compliance

To ensure the eIDAS compliance, there is a list of parameters to specifically set. Those parameters are listed below.

Table 34: eIDAS-Node compliance

Parameter	Resulting value
disallow_self_signed_certificate	True: do not allow self-signed and expired certificates
check_certificate_validity_period	True: do not allow expired certificates
metadata.activate	True: specifies that metadata is generated by the Connector
metadata.restrict.http	True : metadata must be only available via HTTPS
tls.enabled.protocols	TLSv1.1,TLSv1.2: SSL/TLS enabled protocols
metadata.check.signature	True : metadata received from a partner must be signed
metadata.validity.duration	Metadata validity period in seconds. Default=86400 (i.e. one day)
response.encryption.mandatory	True: do not allow response not encrypted
Validate.binding	True: the bindings are validated
security.header.csp.enabled	True: the content-security and security checks are enabled (HSTS, Mozilla directives, X-content-Type-Options, X-frame-options,
disable.check.mandatory.eidas.attributes	False: check the eIDAS minimum dataset constraint. **Note:** this parameter is used by both Proxy Service and Connector.
disable.check.representative.attributes	False: check the eIDAS Request representative rule (must not contain representative attributes). **Note:** this parameter is used by both Proxy Service and Connector.
response.encryption.mandatory	True : check if the response payload is encrypted
check.citizencertificate.serviceCertificate	True : check if the CN of the certificate used for signing the response is the same than the citizen country of the SamlRequest

To ensure compliance, the following checks are made:

- the Level of Assurance indicated in the Assertion matches or exceeds the requested Level of Assurance;

- the Response will not be transmitted to a URL other than the AssertionConsumerServiceURL in the metadata of the eIDAS-Node Connector.

Remark: To improve the resilience of the application, we strongly recommend using the cache instances used for request anti-replay and SAML metadata using Hazelcast services. (please see Appendix D for further details)

7.1. eIDAS-Node configuration recommendations check list for production

To be compliant with the eIDAS regulation, there are things to consider when installing the software in a production environment. These details are more related to the environment and the application server configuration than the configuration of the application itself.

The following check list shows the features to consider:

- Use only **strong** not self-signed certificates;
- Protect your application server from fingerprinting (see section 2.5 — *Setting Software Security Protection*);
- Restrict access to admin interfaces on the production server : Administrator interfaces may be present in the application or on the application server to allow certain users to undertake privileged activities on the site, access needs to be restricted (see your server documentation);
- If the application is not intended to use frames, set the protection against framing by activating the 'X-Frame-Options' header or CSP 'frame-ancestors' directive in the `eidas.xml` file (see sections 13.1.5 — *X-Frame-Options* and 13.1.1 - *Content Security Policy*);
- In the application web.xml, enable secure flag on cookies and HTTP Strict Transport Security (HSTS) header (see section 13.1.4 — *Strict-Transport-Security*);
- Set the logging detail to INFO for audit trails (see section 15 — *eIDAS-Node Error and Event Logging*);
- Configure the anti-replay cache to production (see section 4.1.4.5 - *Additional Configuration — Anti-replay Cache and* Correlation Map Configuration);

The following features should be checked in the *eIDAS Technical Specifications* document as the specifications evolve:

- Configure the application to use the correct version of TLS;
- Use cipher suites that follow the recommendations and provide perfect forward secrecy (PFS);
- Ensure that the algorithms used for signing and encryption are configured in the whitelist and that they conform to the *eIDAS Technical Specifications*.

8. eIDAS Node SAML XML Encryption

8.1. Introduction

This section describes the W3C recommendations and how SAML XML encryption is implemented and integrated in eID.

Encryption of the sensitive data carried in SAML 2.0 Requests and Assertions is discussed alongside the use of AEAD algorithms as essential building blocks.

8.2. Requirement description

The primary requirement of encryption is to protect the citizen against malicious attacks from within the citizen's environment which could:

- disrupt operation
- gather sensitive information from a citizen; or
- gain access to a citizen's environment.

As there is no control of the environment where the citizen operates, this environment cannot be trusted and additional security measures must be taken. For these security measures, only strong, standard algorithms and strong keys are used in line with international standards. You should ensure that effective key management is in place.

For transport confidentiality (eIDAS-Node-to-eIDAS-Node) the following options were proposed:

- End-to-end SAML encryption.

  Please note that this is NOT end-to-end confidentiality between the IdP and the SP, this is end-to-end between eIDAS-Nodes.

- Encryption at the application level should not be imposed; nothing should be imposed on the SP or IdP.

- Data privacy and confidentiality and has been focusing on the scenario where the personal information data is compromised while in the clear (i.e. unencrypted) in the user’s browser.

- The encryption functionality should be easily configurable at the eIDAS-Node level.

  The encryption is enabled or disabled by configuration from an external file to provide the possibility of switching it without rebuilding the application.

8.3. XML 1.1 Encryption Recommendation

The W3C defines a recommendation for XML encryption in XML Encryption Syntax and Processing in which it "*specifies a process for encrypting data and representing the result in XML*".

The current version of the standard is 1.1.

8.4. Overview of supported features

This section describes the features that are supported in this version of eID:

- Encryption Granularity
  - Element (recommended)
  - Character data
  - XML document
- Symmetric Key Encryption

8.5. Encryption granularity

The following sections show examples of degrees of encryption granularity. They show:

- Encryption of an entire element
- Encryption of the content elements of an element
- Encryption of the character content of an element
- Encryption of the entire document

8.6. Encryption of an entire element

In this example the entire `CreditCard` element is encrypted from its start to end tags. The cardholder's `Name` is not considered sensitive and so remains 'in the clear' (i.e. unencrypted).

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000"
Currency="EUR">
    <Number>4019 2445 0277
5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <EncryptedData Type="…" xmlns="…">
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</PaymentInfo>
```

Figure 10: Encryption of an entire element

8.6.1. Encryption of the content elements of an element

In this example the credit card `Number`, `Issuer` and `Expiration` date content elements are encrypted. The cardholder's name, card limit and currency are not considered sensitive and so remain unencrypted.

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000"
Currency="EUR">
    <Number>4019 2445 0277
5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000" Currency="EUR">
    <EncryptedData Type="…" xmlns="…">
      <CipherData>

<CipherValue>A23B45C56</CipherValue>
      </CipherData>
    </EncryptedData>
  </CreditCard>
</PaymentInfo>
```

Figure 11: Encryption of the content elements of an element

8.6.2. Encryption of the character content of an element

In this example only the character content of the card's `Number` is encrypted.

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000"
Currency="EUR">
    <Number>4019 2445 0277
5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000" Currency="EUR">
    <Number>
      <EncryptedData Type="…" xmlns="…">
        <CipherData>

<CipherValue>A23B45C56</CipherValue>
        </CipherData>
      </EncryptedData>
    </Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figure 12: Encryption of the character content of an element

8.6.3. Encryption of the entire document

In this example the entire document is encrypted.

```
<PaymentInfo xmlns="…">
  <Name>John Smith</Name>
  <CreditCard Limit="5,000"
Currency="EUR">
    <Number>4019 2445 0277
5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

```
<EncryptedData Type="…" xmlns="…"
MimeType="text/xml">
  <CipherData>
    <CipherValue>A23B45C56</CipherValue>
  </CipherData>
</EncryptedData>
```

Figure 13: Encryption of the entire document

8.6.4. Symmetric key encryption

Shared secret key encryption algorithms are especially specified for encrypting and decrypting symmetric keys.

They appear as `Algorithm` attribute values (A) to `EncryptionMethod` elements (B).

They are children of `EncryptedKey` (C) which is in turn a child of `ds:KeyInfo` (D) which is in turn a child of `EncryptedData` (E) (or another `EncryptedKey`).

```
<saml2:EncryptedAssertion>

E       <xenc:EncryptedData Id="_6a47298ff6092f82d9e540479c46bdfb "
        Type="http://www.w3.org/2001/04/xmlenc#Element">

          <xenc:EncryptionMethod

          Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />

D         <ds:KeyInfo>

C             <xenc:EncryptedKey Id="_74229d1928a63480c0895c79497d20fe">
B               <xenc:EncryptionMethod
A               Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                  <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
                </xenc:EncryptionMethod>

                <ds:KeyInfo>
                 <ds:X509Data>

<ds:X509Certificate>MIIDJzC...8mYfX8/jw==</ds:X509Certificate>
                 </ds:X509Data>
                </ds:KeyInfo>
```

```
                <xenc:CipherData>
                   <xenc:CipherValue>bSYaL0cXyC...UjDBQ==</xenc:CipherValue>
                </xenc:CipherData>
             </xenc:EncryptedKey>


          </ds:KeyInfo>


          <xenc:CipherData>
             <xenc:CipherValue>pTvKqOqq...kfBb1rw=</xenc:CipherValue>
          </xenc:CipherData>
       </xenc:EncryptedData>
    </saml2:EncryptedAssertion>
```

For further information, refer to http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/Overview.html#sec-Usage.

The following steps show the process when encrypting with one-time-use symmetric key.

1. Start XML encryption.
2. Retrieve recipient's existing Asymmetric Public key.
3. Create new Symmetric key.
4. Encrypt XML with Symmetric key.
5. Encrypt Symmetric key with RSA using Public key.
6. Include encrypted Symmetric key into XML with EncryptedKey element.
7. Send XML.

8.7. SAML 2.0 AuthnResponse Assertion Encryption

The standard way of encrypting the SAML Response is to encrypt the Assertions within as described below.

8.7.1. Assertion encryption support by SAML 2.0

The common type for storing encrypted data is the `EncryptedElementType`. This type is inherited by the `EncryptedAssertion` element of SAML.

8.7.2. Pseudo implementation of encryption of SAML Response

1. Check if the functionality is enabled.
2. Set the appropriate data encryption and key encryption algorithms (see section 8.8.4 — *Code snippet – Data & key encryption parameters*.
3. Acquire certificate of the relaying party for symmetric key encryption and use as the key encryption credential (see section 8.8.3 — *Code snippet – Certification credential for encryption*).
4. Clone SAML Response instance to avoid in-place modification of original SAML Response.
5. Generate new symmetric key.
6. Set the KeyPlacement INLINE (see section 8.8.5 — *Code snippet – Set up open SAML* encrypter)
7. For each Assertion in the SAML Response:
    a. Encrypt assertion.
    b. Add the encrypted assertion to the EncyptedAssertions list of the response.

    (see section *8.8.6 — Code Snippet – Assertion* encryption)
8. Clear all plain Assertion elements from the SAML Response.
9. Return the cloned, encrypted SAML Response (only the SAML Assertion is encrypted).

8.7.3. Pseudo implementation of decryption of SAML Response

1. Check if functionality is enabled.
2. Acquire the single KeyInfo of SAML Response if present.(see section 8.8.8 – Code snippet – Locate & construct the single certificate for decryption in the SAML Response)
3. Extract X509Certificate from the single KeyInfo.
4. Clone SAML Response instance to avoid in-place modification of original SAML Response.
5. Find the appropriate KeyStore entry based on the extracted certificate and retrieve the PrivateKey (see section 8.8.9 – Code snippet – Credential based on the certification for decryption)
6. For each EncryptedAssertion in the SAML Response:
    a. Acquire EncryptedKey (symmetric) from the KeyInfo of the EncryptedAssertion.
    b. Decrypt symmetric key with the PrivateKey.
    c. With the decrypted symmetric SecretKey decrypt the EncryptedAssertion instance. (see section 8.8.10 – Code snippet – Assertion decryption)
    d. Add the decrypted Assertion to the Assertions list of the response.
7. Clear all plain EncryptedAssertion elements from the SAML Response.

8. Return the cloned, decrypted SAML Response.

8.7.4. Encryption configuration

The encryption can be configured globally for all instances or individually for each instance, so, depending on your needs, it may differ from the one shown here.

The first step is to add into the `SAMLEngine.xml` for each module a new configuration for each instance to indicate the configuration files that will be used

```
  <!-- Settings module encryption -->
    <configuration name="EncryptionConf">
       <!-- Specific signature module -->
       <parameter name="class"
           value="eu.eidas.auth.engine.core.impl.EncryptionSW" />
           <!-- Settings specific module
             responseTo/FromPointAlias & requestTo/FromPointAlias parameters
will be added -->
       <parameter name="fileConfiguration" value="EncryptModule_Service.xml" />
    </configuration>
```

The `fileConfiguration` parameter defines the file that contains:

- the keystore,
- the certificates for each country used to encrypt the SAML Responses,
- its own certificate; and
- the relative path to the external file that configures the activation of the encryption for each country.
- This file will be usually allocated inside the module along with the `SignModule_<instance>.xml` files. There will be one per instance.

Inside the file you need to define a `keyStore` with the trusted certificates for each country.

The property `encryptionActivation` defines the file containing the flags to activate the encryption for the countries. This can be one single file or one file per instance depending on your needs and its complexity. This file is intended to be placed outside the module so configuration changes can be made without rebuilding the whole application.

It contains one property for each country that defines if the Response sent to the country should be encrypted. The decryption process is automatically managed. If the received response is encrypted then it decrypts it, otherwise does nothing.

`EncryptTo.<CountryCode>`: Determines if the response sent to the country defined by its country code must be encrypted.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<!-- Activate encryption in the module -->
       <entry key="Encryption.enable">true</entry>
```

```
<entry key="EncryptTo.CA">false</entry>
<entry key="EncryptTo.CB">false</entry>
<entry key="EncryptTo.CC">false</entry>
<entry key="EncryptTo.CD">false</entry>
<entry key="EncryptTo.CF">false</entry>
</properties>
```

If some of these parameters are not present the application will work as if the encryption is not activated.

The property `Encryption.enable` is maintained to activate the encryption in the instance itself.

Notes:

1. If encryption and metadata are enabled, the metadata will expose encryption information (see section *9 — eIDAS-Node SAML metadata* for more details about metadata). The public key retrieved from metadata will be used for encryption (instead of the one available locally in the keystore).

2. When the configuration parameter **`response.encryption.mandatory`** is set to true (in `eidas.xml`) then the settings in `encryptionConf.xml` are ignored and each response is either encrypted or an error is raised. Error responses are allowed to be unencrypted.

The diagram below shows a possible example configuration.

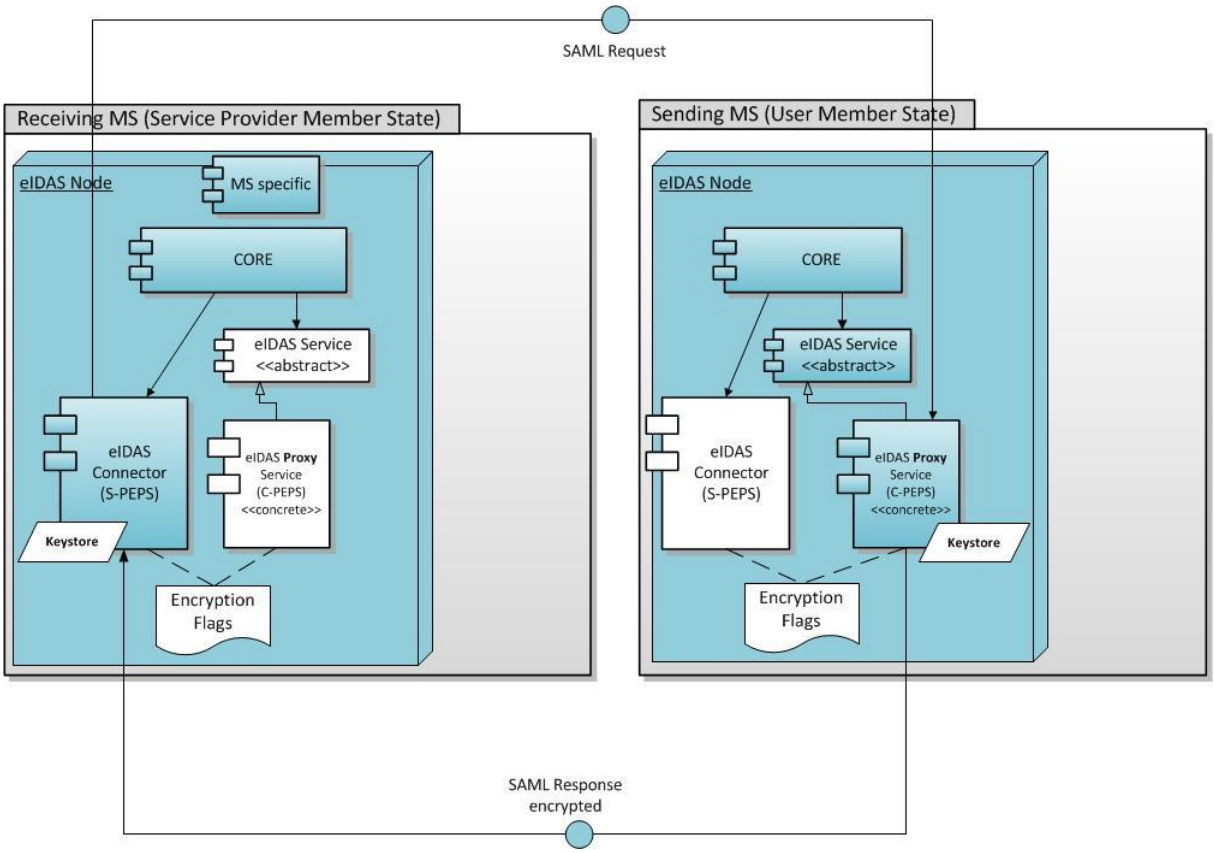


Figure 14: The architecture of SAML encryption

8.7.5. eIDAS SAML 2.0 Encryption example

The following shows an example of encrypting an assertion by replacing plain Assertion with `EncryptedAssertion`.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:eidas="http://eidas.europa.eu/saml-extensions"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained" Destination="http://vs-
cis-k2:8081/SP/ReturnPage" ID="_fb4aed821dbb327caf2aa310a6f877bb"
InResponseTo="_cd5ad7ff3a77529079b39e073597fbc9" IssueInstant="2014-11-
20T12:20:00.319Z" Version="2.0">
   <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:entity">http://eidas-connector.gov.xx</saml2:Issuer>
   <saml2p:Status>…</saml2p:Status>
   <saml2:Assertion ID="_6a47298ff6092f82d9e540479c46bdfb"
      IssueInstant="2014-11-20T12:20:00.334Z" Version="2.0">
      <saml2:Issuer
      Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
         http://eidas-connector.gov.xx
      </saml2:Issuer>
      <saml2:Subject>…</saml2:Subject>
      <saml2:Conditions NotBefore="2014-11-20T12:20:00.334Z"
      NotOnOrAfter="2014-11-20T12:25:00.319Z">…</saml2:Conditions>
      <saml2:AuthnStatement AuthnInstant="2014-11-20T12:20:00.334Z">
```

```
         <saml2:SubjectLocality Address="158.168.60.142" />
         <saml2:AuthnContext>
            <saml2:AuthnContextDecl />
         </saml2:AuthnContext>
      </saml2:AuthnStatement>
      <saml2:AttributeStatement>
         <saml2:Attribute Name=http://eidas.europa.eu/1.0/eIdentifier
         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
            <saml2:AttributeValue
            xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
            xsi:type="xs:anyType">CA/CA/12345</saml2:AttributeValue>
         </saml2:Attribute>
         <saml2:Attribute Name=http://eidas.europa.eu/1.0/givenName
         NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
            <saml2:AttributeValue
            xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
            xsi:type="xs:anyType">Javier</saml2:AttributeValue>
         </saml2:Attribute>
         …
      </saml2:AttributeStatement>
   </saml2:Assertion>
   <saml2:EncryptedAssertion>
      <xenc:EncryptedData Id="_6a47298ff6092f82d9e540479c46bdfb "
      Type="http://www.w3.org/2001/04/xmlenc#Element">
         <xenc:EncryptionMethod
         Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
         <ds:KeyInfo>
            <xenc:EncryptedKey Id="_74229d1928a63480c0895c79497d20fe">
               <xenc:EncryptionMethod
               Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                  <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
               </xenc:EncryptionMethod>
               <ds:KeyInfo>
                  <ds:X509Data>

<ds:X509Certificate>MIIDJzC...8mYfX8/jw==</ds:X509Certificate>
                  </ds:X509Data>
               </ds:KeyInfo>
               <xenc:CipherData>
                  <xenc:CipherValue>bSYaL0cXyC...UjDBQ==</xenc:CipherValue>
               </xenc:CipherData>
            </xenc:EncryptedKey>
         </ds:KeyInfo>
         <xenc:CipherData>
            <xenc:CipherValue>pTvKqOqq...kfBb1rw=</xenc:CipherValue>
         </xenc:CipherData>
      </xenc:EncryptedData>
   </saml2:EncryptedAssertion>
</saml2p:Response>
```

8.7.6. eIDAS SAML 2.0 Encryption and Signature

EIDAS and STORK use XML Signature to verify the authenticity of the SAML messages.

The following rule must apply to the Encryption implementation:

- The signature must be created using the encrypted SAML message.
- The verification of signature must be executed on the encrypted SAML message

It means that the following pseudo process must be implemented:

1. Construct SAML Response.
2. Encrypt SAML Response.
3. Sign Encrypted SAML Response (only the SAML Assertion is encrypted).
4. Send Encrypted SAML Response (only the SAML Assertion is encrypted) to relaying party.
5. Verify the signature of Encrypted SAML Response (only the SAML Assertion is encrypted) at the relaying party.
6. If success then decrypt SAML Assertion
7. Process decrypted SAML Response

8.7.7. eIDAS SAML 2.0 Encryption with Signature example

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:eidas="http://eidas.europa.eu/saml-extensions"
Consent="urn:oasis:names:tc:SAML:2.0:consent:obtained"
Destination="http://localhost:8080/eidasNode/SpecificIdPResponse"
ID="_f38631b536398fb5e9432e91bb7f764d"
InResponseTo="_9884b5a27102a5870455be919d126faa" IssueInstant="2014-12-
23T10:32:12.751Z" Version="2.0">
   <saml2:Issuer
   Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
        http://eidas-connector.gov.xx
   </saml2:Issuer>
   <ds:Signature>
      <ds:SignedInfo>
         <ds:CanonicalizationMethod
         Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
         <ds:SignatureMethod
         Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
         <ds:Reference URI="#_f38631b536398fb5e9432e91bb7f764d">
            <ds:Transforms>
               <ds:Transform
               Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"
/>
               <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
/>
            <ds:DigestValue>bcWvl4NOSgKmVZglli9SKRKJipE=</ds:DigestValue>
         </ds:Reference>
      </ds:SignedInfo>
      <ds:SignatureValue>PdwB/…IW+yg==</ds:SignatureValue>
      <ds:KeyInfo>
         <ds:X509Data>
           <ds:X509Certificate>MIIDJzC…YfX8/jw==</ds:X509Certificate>
         </ds:X509Data>
      </ds:KeyInfo>
   </ds:Signature>
   <saml2p:Status>
      <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success" />
      <saml2p:StatusMessage>
          urn:oasis:names:tc:SAML:2.0:status:Success
      </saml2p:StatusMessage>
```

```
   </saml2p:Status>
   <saml2:EncryptedAssertion>
      <xenc:EncryptedData xmlns:xenc=http://www.w3.org/2001/04/xmlenc#
      Id="_485c8629476d743eb85c244ac3f113f0"
      Type="http://www.w3.org/2001/04/xmlenc#Element">
         <xenc:EncryptionMethod
         Algorithm="http://www.w3.org/2009/xmlenc11#aes256-gcm" />
         <ds:KeyInfo>
            <xenc:EncryptedKey Id="_74229d1928a63480c0895c79497d20fe">
               <xenc:EncryptionMethod
               Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p">
                  <ds:DigestMethod
                  Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
               </xenc:EncryptionMethod>
               <ds:KeyInfo>
                  <ds:X509Data>
                    <ds:X509Certificate>MII…8mYfX8/jw==</ds:X509Certificate>
                  </ds:X509Data>
               </ds:KeyInfo>
               <xenc:CipherData>
                  <xenc:CipherValue>bS…jDBQ==</xenc:CipherValue>
               </xenc:CipherData>
            </xenc:EncryptedKey>
         </ds:KeyInfo>
         <xenc:CipherData>
            <xenc:CipherValue>pTvKq…Bb1rw=</xenc:CipherValue>
         </xenc:CipherData>
      </xenc:EncryptedData>
   </saml2:EncryptedAssertion>
</saml2p:Response>
```

8.8. XML Encryption/Decryption implementation

8.8.1. OpenSAML - XML Tooling

Currently eIDAS uses OpenSAML signature utilities. The XML Encryption/Decryption engine of Open SAML named xmltooling implements the necessary AES-GCM algorithms:
http://www.w3.org/2009/xmlenc11#aes256-gcm is strongly recommended to use.

In order to be able to implement large key size encryption refer to grepcode.com for the details of implemented algorithms:
http://grepcode.com/file/repo1.maven.org/maven2/org.opensaml/xmltooling/1.4.1/org/opensaml/xml/encryption/EncryptionConstants.java?av=f

For implementation details refer to:
https://wiki.shibboleth.net/confluence/display/OpenSAML/OSTwoUserManJavaXMLEncryption

An advantage of xmltooling is that it provides SAML Specific Encrypter and Decrypter for Assertions besides the generic XML Encryption. **The generic XML Encryption of xmltooling can be used.**

8.8.2. Component dependencies

The following diagram illustrates the dependencies of the various components in an eIDAS implementation.

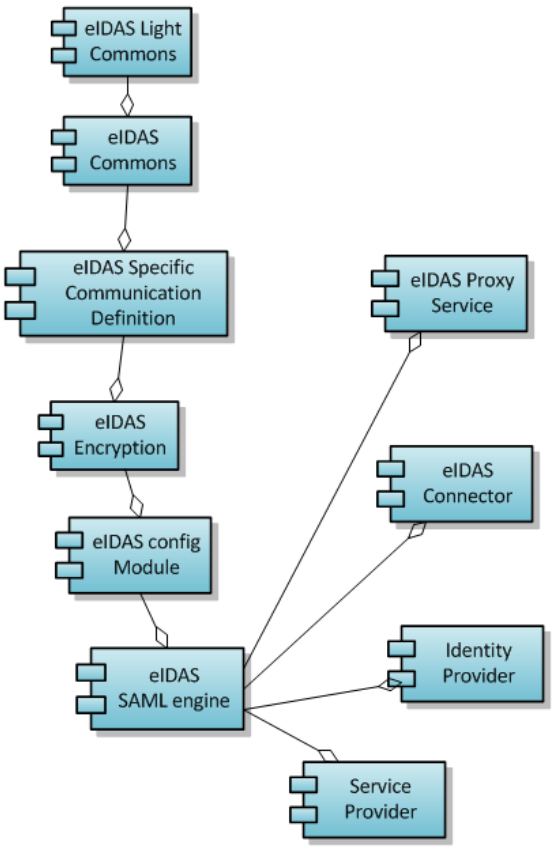


Figure 15: Component dependencies

8.8.3. Code snippet – Certification credential for encryption

```
samlAuthnResponseEncrypter = new SAMLAuthnResponseEncrypter();
…
// Load certificate matching the configured alias
X509Certificate responsePointAliasCert = (X509Certificate)
encryptionKeyStore.getCertificate(alias);

// Create basic credential and set the EntityCertificate
BasicX509Credential credential = new BasicX509Credential();
credential.setEntityCertificate(responsePointAliasCert);

// Execute encryption
samlAuthnResponseEncrypter.encryptSAMLResponse(authResponse, credential);
```

8.8.4. Code snippet – Data & key encryption parameters

```
// Set Data Encryption parameters
EncryptionParameters encParams = new EncryptionParameters();
encParams.setAlgorithm(…);

// Set Key Encryption parameters
KeyEncryptionParameters kekParams = new KeyEncryptionParameters();
kekParams.setEncryptionCredential(credential);
kekParams.setAlgorithm(…);
KeyInfoGeneratorFactory kigf =
        Configuration.getGlobalSecurityConfiguration()
                .getKeyInfoGeneratorManager().getDefaultManager()
                .getFactory(credential);
kekParams.setKeyInfoGenerator(kigf.newInstance());
```

8.8.5. Code snippet – Set up open SAML encrypter

```
// Setup Open SAML Encrypter
Encrypter samlEncrypter = new Encrypter(encParams, kekParams);
samlEncrypter.setKeyPlacement(Encrypter.KeyPlacement.INLINE);
```

8.8.6. Code Snippet – Assertion encryption

```
samlResponseEncryptee = XMLObjectHelper.cloneXMLObject(samlResponse);
…
for (Assertion assertion : samlResponseEncryptee.getAssertions()) {
    manageNamespaces(assertion);
    EncryptedAssertion encryptedAssertion = samlEncrypter.encrypt(assertion);
    samlResponseEncryptee.getEncryptedAssertions().add(encryptedAssertion);
}
samlResponseEncryptee.getAssertions().clear();
```

8.8.7. Code snippet – Manage specific namespace prefix

Necessary when the SAML Response XML does not use the "saml:" prefix. In this case the decryption will not be able to identify the unknown prefix of the detached Assertion element unless the following is applied!

```
Set<Namespace> namespaces = assertion.getNamespaceManager().getNamespaces();
for (Namespace namespace : namespaces) {
    if
("urn:oasis:names:tc:SAML:2.0:assertion".equals(namespace.getNamespaceURI())
            && assertion.getDOM().getAttributeNode("xmlns:" +
                    namespace.getNamespacePrefix()) == null) {
        assertion.getNamespaceManager().registerNamespaceDeclaration(namespace);
        assertion.getDOM().setAttribute("xmlns:" +
                    namespace.getNamespacePrefix(), namespace.getNamespaceURI());
    }
}
```

8.8.8. Code snippet – Locate & construct the single certificate for decryption in the SAML Response

The following snippet presumes that the encryption was applied as the above configuration shows.

```
EncryptedAssertion encAssertion = authResponse.getEncryptedAssertions().get(0);
EncryptedKey encryptedSymmetricKey = encAssertion.getEncryptedData().
        getKeyInfo().getEncryptedKeys().get(0);
org.opensaml.xml.signature.X509Certificate
    keyInfoX509Cert = encryptedSymmetricKey.
        getKeyInfo().getX509Datas().get(0).getX509Certificates().get(0);
final ByteArrayInputStream bis = new ByteArrayInputStream(Base64
        .decode(keyInfoX509Cert.getValue()));
final CertificateFactory certFact = CertificateFactory
        .getInstance("X.509");
final X509Certificate keyInfoCert = (X509Certificate) certFact
        .generateCertificate(bis);
```

8.8.9. Code snippet – Credential based on the certification for decryption

```
samlAuthnResponseDecrypter = new SAMLAuthnResponseDecrypter();
for (final Enumeration<String> e = encryptionKeyStore.aliases();
                e.hasMoreElements(); ) {
    aliasCert = e.nextElement();
    responsePointAliasCert = (X509Certificate) encryptionKeyStore.
        getCertificate(aliasCert);
    //Check if certificates equal
    if(Arrays.equals(keyInfoCert.getTBSCertificate(),
              responsePointAliasCert.getTBSCertificate())) {
        alias = aliasCert;
        break;
    }
}
//Handle if certificate not found… etc.
…
//Get PrivateKey by found alias
final PrivateKey responsePointAliasPrivateKey = (PrivateKey)
encryptionKeyStore.getKey(
        alias, properties.getProperty("keyPassword").toCharArray());

// Create basic credential and set the PrivateKey
BasicX509Credential credential = new BasicX509Credential();
credential.setPrivateKey(responsePointAliasPrivateKey);

// Execute decryption
samlAuthnResponseDecrypter.decryptSAMLResponse(authResponse, credential);
```

8.8.10. Code snippet – Assertion decryption

```
samlResponseDecryptee = XMLObjectHelper.cloneXMLObject(samlResponseEncrypted);
…
for (EncryptedAssertion encAssertion :
          samlResponseDecryptee.getEncryptedAssertions()) {
    EncryptedKey encryptedSymmetricKey =

encAssertion.getEncryptedData().getKeyInfo().getEncryptedKeys().get(0);

    //Key Decrypter
    Decrypter keyDecrypter = new Decrypter(null,
           new StaticKeyInfoCredentialResolver(credential), null);
    SecretKey dataDecKey = (SecretKey) keyDecrypter.decryptKey(
           encryptedSymmetricKey,

encAssertion.getEncryptedData().getEncryptionMethod().getAlgorithm());

    //Data Decrypter
    Credential dataDecCredential =
SecurityHelper.getSimpleCredential(dataDecKey);
    Decrypter dataDecrypter = new Decrypter(
            new StaticKeyInfoCredentialResolver(dataDecCredential), null, null);
    dataDecrypter.setRootInNewDocument(true);
    Assertion assertion = dataDecrypter.decrypt(encAssertion);
    samlResponseDecryptee.getAssertions().add(assertion);
}
samlResponseDecryptee.getEncryptedAssertions().clear();
```

8.9. Sources of further information

The following sources provide further information on the SAML XML encryption standard and its implementation.

Symmetric key encryption

http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p with 2048 key size.
Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files must be applied (refer to the download site of JCE Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 Download at
http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

XMLEnc XSD

http://www.w3.org/TR/2013/PR-xmlenc-core1-20130124/xenc-schema.xsd

9. eIDAS-Node SAML metadata

9.1. Presentation

SAML metadata are configuration data required to automatically negotiate agreements between system entities, comprising:

- Identifiers;
- binding support and endpoints;
- certificates;
- keys;
- cryptographic capabilities;
- security and privacy policies.

SAML metadata is standardised by OASIS and signed XML data (signing SAML metadata is not mandatory but is assumed in this document). SAML can be provided as a metadata file at a URL, as indirect resolution through DNS, or by other means.

The advantage of SAML metadata is that it is part of the SAML specifications, thus already closely related to the planned eIDAS interoperability standard using SAML as message format and exchange protocol. It is supported by SAML implementations and libraries.

The infrastructure data needed is twofold:

- notification-related and security data e.g. authentication online
- sole technical data e.g. supported protocol versions.

Both are needed for a functioning infrastructure. The distinction above is made, as the issuance and security relevance may be different. The former (notification-related data) is information provided by the MS that also relates to parties being liable under eIDAS art. 11. The latter (sole technical data) relates to the functioning of components.

See Table 35 for an indicative list of metadata.

9.2. Use cases

9.2.1. Identification of eIDAS-Nodes

To provide an uninterrupted chain of trust for authentications, as well as an uninterrupted chain of responsibility for integrity/authenticity and confidentiality for personal identification data, eIDAS-Nodes MUST be securely identified before transmitting data to them or accepting data from them.

9.2.1.1. Proxy-based schemes

Certificates for SAML signing and encryption of messages between Connector and Proxy Service are exchanged via SAML Metadata.

9.2.1.2. Middleware-based schemes

Certificates for SAML signing and encryption of messages between an eIDAS-Node Connector and eIDAS-Middleware Services are exchanged directly between the entities, since there is a one-to-one correspondence between an eIDAS-Node Connector and eIDAS Middleware-Service.

9.2.2. Request messages verification

Each eIDAS-Service MUST verify the integrity/authenticity of a SAML Request message before processing the request.

For eIDAS-Proxy-Services, this comprises the following steps:

1. Retrieve the SAML Metadata object of the sender from the metadata URL contained in the request or from a local cache. A local cache is used to reduce latency (redirect processing is usually faster than POST-processing) and to enhance usability in environments where JavaScript is not available, e.g. corporate networks.
2. Verify the signature of the SAML Metadata object including Certification Path Validation according to [RFC5280]. The Path MUST start at a valid trust anchor.
3. Extract the signature certificate of the sender from the SAML Metadata and use them to verify the signature of the SAML Request message.

Request messages which cannot be verified via this procedure MUST be rejected.

9.2.3. Response messages verification

Each eIDAS-Connector MUST verify the authenticity of a SAML Response message before processing the included assertion.

For messages originating at an eIDAS-Proxy Service, this comprises the following steps:

4. Retrieve the SAML Metadata object of the sender, either from the metadata URL contained in the Assertion or from a local cache. A local cache is used to reduce latency (redirect processing is usually faster than POST-processing) and to enhance usability in environments where JavaScript is not available, e.g. corporate networks.
5. Verify the signature of the SAML Metadata object including Certification Path Validation according to [RFC5280]. The Path MUST start at a valid trust anchor.
6. Extract the signature certificate of the sender from the SAML Metadata and using it to verify the signature of the SAML Response message.
7. If the SAML Assertion is signed, its signature MAY be verified.

Assertion messages which cannot be verified via this procedure MUST be rejected. Unsolicited Response Messages MUST NOT be accepted.

See Table 36 for a list of metadata related parameters.

9.2.4. Metadata exchange

To provide an uninterrupted chain of trust for authentications, as well as an uninterrupted chain of responsibility for integrity/authenticity and confidentiality for personal identification data, eIDAS-Nodes must be securely identified. The Architecture defines two communication relationships:

- Communication between eIDAS Connectors and Proxy-Services
- Communication between eIDAS Connectors and Middleware-Services.

For Middleware-Services, there is a one-to-one relationship between the Connector and the Middleware-Service; identification is done directly, e.g. via self-signed certificates.

Therefore this section is only concerned with exchanging metadata for Connectors and Proxy-Services, although direct exchange can also be done using the same mechanism.

Metadata exchange is based on the following principles:

- The trust anchors for all eIDAS-Nodes are the MS's. No central trust anchor is provided. Trust Anchors are exchanged bilaterally between MS's.
- Metadata are distributed in the form of SAML Metadata [SAML-Meta]. Metadata objects are signed by the Trust Anchor or by another entity (e.g. the operator of the eIDAS-Node) authorised via a certificate chain starting from the trust anchor.

9.2.4.1. Trust anchor

All MSs MUST bilaterally exchange trust anchors in the form of certificates, certifying a signing key held by the MS (the 'Root'). This signing key can either be used:

- to directly sign SAML metadata objects, or
- as root certificate of a PKI used to sign SAML metadata objects.

Certificates of the root and subordinate certificates SHALL follow [RFC5280].

MS MUST ensure that all SAML metadata objects signed directly or indirectly under this Root describe valid eIDAS-Nodes established in that MS.

9.2.4.2. SAML metadata

The eIDAS-Connector provides metadata about the Connector/Proxy Service in the form of SAML Metadata (see *SAML-Meta* in section *9.5 — References*).

SAML Metadata objects are signed and include a certificate chain starting at a trust anchor and terminating with a certificate certifying the key used to sign the Metadata object.

The SAML Metadata Interoperability Profile Version MUST be followed (see *SAML-MetaIOP* in section *9.5 — References*]). Certificates MUST be encapsulated in X509Certificate-elements.

The current version of the eIDAS-Node publishes metadata information under the `EntityDescriptor` element. It may read metadata information available under either:

- EntityDescriptor element; or
- `EntitiesDescriptor` element data from a static file (aggregate metadata).

Note: SAML metadata are not needed for eIDAS-Middleware-Services, since certificates are exchanged directly between eIDAS-Connector and corresponding eIDAS-Middleware-Services.

9.2.4.3. Metadata location

The SAML Metadata are only available under an HTTPS URL.

SAML Requests and Responses contain an HTTPS URL in the <`Issuer`> element pointing to the SAML metadata object of the issuer of the request/assertion (see section *4.1.1 of SAML-Meta* 'Well-Known Location' method in section *9.5 — References*).

9.2.4.4. Metadata verification

For verification of SAML metadata, verifiers build and verify a certificate path according to RFC5280 (in section *9.5 — References*) starting from a trusted Trust Anchor (see section *9.2.4.1 — Trust anchor*) and ending at the signer of the metadata object. Revocation checks are performed for all certificates containing revocation information.

All restrictions contained in the metadata object (e.g. validity period) are honoured by the verifier.

9.3. Message format

9.3.1. Metadata in SAML Requests & SAML Responses

SAML Requests and Assertions (SAML Responses) contain an HTTPS URL in the `<Issuer>` element pointing to the SAML Metadata object of the issuer of the request/assertion (see section *4.1.1 of SAML-Meta*: Metadata for the OASIS Security Assertion Markup Language at https://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf).

9.3.2. Metadata profile for eIDAS-Nodes

The profile contains the following information:

- the MS operating the eIDAS-Node;
- the URL under which the eIDAS-Node is operated;
- a communication address (preferably email);
- the certificates corresponding to the SAML signature and decryption keys;
- an indication if the eIDAS-Node serves public and/or private parties.

9.3.3. List of eIDAS metadata

Table 35: List of eIDAS metadata

Metadata	SAML MD	Comment
Validity period	`validUntil`	Indicates expiration date of metadata elements
Protocol version(s)	`protocolSupport-Enumeration`	For multi-protocol support, this is likely to change and be amended over time
Supported NameID	`SSODescriptor NameIDFormat`	In case several NameID formats can get requested
Supported attributes	`SSODescriptor Attribute; EntityAttribute extensions`	At least the minimum data set and available matching data. Should also allow for indicating additional sector-specific attributes.
SAML signer	`KeyDescriptor`	Certificate to sign SAML requests or responses
Encryption certificate	`KeyDescriptor`	Certificate to encrypt a SAML Response
Supported LoA	`SAML Extensions EntityAttributes AttributeName http://eidas.europa.eu/LoA`	Automatically retrieving which LoAs are supported allows a component (relying party, eIDAS-Node Connector, V-IDP) to only redirect to MS that also offer the LoA needed by a relying party
Organization info	`OrganizationName, organizationDisplayName, OrganizationURL`	Organization responsible for an entity
Contact info	`ContactPersonType, ContactPersonGivenName, ContactPersonSurname, ContactPersonEmail, ContactPersonPhone`	Support contact for an entity (e.g. eIDAS-Node, V-IDP)

9.4. Details of the metadata used in the eIDAS-Node

The following validations are made on the metadata (when receiving a message):

- verify validity duration (see `metadata.validity.duration` parameter);
- check the metadata signature (trusted certificate loaded in the keystore – see `metadata.check.signature` parameter);
- check the `assertionConsumerServiceURL` is the same between the metadata and the message;

9.4.1. Support of dynamic and cached use of metadata

The switch `metadata.http.retrieval` activates the dynamic retrieval of metadata information using http/https.

If the dynamic retrieval is disabled, the `metadata.file.repository` parameter needs to contain the location (folder) where static SAML metadata configuration is stored.

The application will process all the files contained in the folder. If the folder does not exist or the parameter refers to an unavailable location, only dynamically retrieved metadata may be used.

If the static metadata is expired, the application will try to reach the remote location.

Table 36: Metadata related parameters

Key	Description
`metadata.file.repository`	Location (folder) where static SAML metadata configuration is stored. If it does not exist or refers an unavailable location, only dynamically retrieved metadata may be used.
`metadata.http.retrieval`	Activate the retrieval of metadata information using http/https. Default is 'true'. When set on 'false', only static metadata may be used.
`metadata.restrict.http`	Disable http for metadata retrieval (such that only https will be allowed)
`tls.enabled.protocols`	The SSL/TLS protocols to be used when retrieving metadata via https. The default values, as by eIDAS specification, are: TLSv1.1,TLSv1.2.
`metadata.sector`	Value of SPType (public or private) to be published in the metadata (see *eIDAS Technical Specifications)*
`metadata.check.signature`	When set to 'false', the check of metadata signature is disabled. (Note: by default metadata signature validation is active)
`metadata.validity.duration`	Duration of validity for dynamic metadata (in seconds). Default is 86400 (one day)

For SP and IDP metadata, please check installation settings (`sp.properties` and `idp.properties`), because static and http metadata settings differ from the ones used in the eIDAS-Node.

9.4.2. Internal cache behaviour

When a piece of metadata is requested for processing (e.g. when an incoming request is processed by eIDAS), the validity is checked. An error will be printed in the logs if the metadata has expired.

A statically loaded metadata (if `metadata.http.retrieval` set to false) will not be replaced by a piece of metadata retrieved through http/https.

9.4.3. Paramatrisation of the metadata signing certificate

The certificate used to sign SAML messages is not the same as that used to sign the metadata. All MS MUST bilaterally exchange trust anchors in the form of certificates, certifying a signing key held by the MS (the "Root"). This signing key can either be used to directly sign SAML metadata objects, or as root certificate of a PKI used to sign SAML metadata objects.

Each communication point in the eIDAS node needs to be configured to use a specific certificate to sign its metadata. It is configured into the same file used to configure the encryption (see section 10.3 — *Configuration* for signature configuration information).This file will be usually allocated inside the module along with the `SignModule_<instance>.xml` files. This file should contain the following lines:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
	<comment>SWModule sign with JKS.</comment>
	<entry key="keyStorePath">/keystores/keycountry1.jks</entry>
	<entry key="keyStorePassword">passkey1</entry>
	<entry key="keyPassword">pass1</entry>
	<entry key="issuer">CN=eidas, C=ES</entry>
```



```
        <entry key="serialNumber">4BA0BD66</entry>
        <entry key="keyStoreType">JKS</entry>
        <!—Metadata signature configuration -->
        <entry key="metadata.keyStorePath">eidasKeyStore_METADATA.jks</entry>
        <entry key="metadata.keyStorePassword">keystorepassword</entry>
        <entry key="metadata.keyPassword">keypassword</entry>
        <entry key="metadata.issuer">issuerDN</entry>
        <entry key="metadata.serialNumber">serialNumber</entry>
        <entry key="metadata.keyStoreType">JKS</entry>
</properties>
```

If a parameter is missing, the corresponding default parameter's value will be used. E.g. if `metadata.keyStorePath` is not set, then the value from `keyStorePath` parameter is used.

9.5. References

- *eIDAS CryptoeIDAS*: Security Requirements for TLS and SAML
- *eIDAS Interop*: Interoperability Architecture
- *RFC5280IETF: RFC 5280*: D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk: RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
- *SAML-CoreOASIS*: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0
- *SAML-BindingOASIS*: Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0
- *SAML-SecOASIS*: F. Hirsch, R. Philpott, E. Maler: Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0
- *SAML-MetaOASIS*: Metadata for the OASIS Security Assertion Markup Language (SAML) v2.0
- *SAML-Meta: OASIS*: Metadata for the OASIS Security Assertion Markup Language (SAML) v2.0
  http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf
- *SAML-MetaIOP*: OASIS: SAML V2.0 Metadata Interoperability Profile Version 1.0
  https://www.oasis-open.org/committees/download.php/36645/draft-sstc-metadata-iop-2.0-01.pdf
- *SAML-MetaIOPOASIS*: SAML V2.0 Metadata Interoperability Profile Version 1.0
- *XMLSig BPW3C*: XML Signature Best Practices, http://www.w3.org/TR/xmldsig-bestpractices

10. eIDAS-Node SAML Engine

10.1. Introduction

The `saml-engine-X.X.jar` library is a module from the eIDAS project and is used by any module that needs to generate or validate SAML messages with eIDAS specification.

10.2. Dependencies

This library is based on OpenSAML2 (version 2.6.5).

Additionally, it is dependent on `eidas-commons-XX.jar`, `eidas-encryption` and `eidas-configmodule`.

10.3. Configuration

All eIDAS modules that use the SAML engine could create different instances of the engine.

`SAMLEngine.xml`: configuration file, located either by `EIDAS_CONFIG_REPOSITORY` or by `SPECIFIC_CONFIG_REPOSITORY` where the instances are defined:

```
<!-- Configuration name-->
<instance name="CONF1">
 <!-- Configurations parameters SamlEngine  -->
 <configuration name="SamlEngineConf">
   <parameter name="fileConfiguration" value="SamlEngine_Conf1.xml" />
 </configuration>

 <!-- Settings module signature-->
 <configuration name="SignatureConf">
   <!-- Specific signature module -->
   <parameter name="class" value="eu.eidas.auth.engine.core.impl.SignSW" />
   <!-- Settings specific module -->
   <parameter name="fileConfiguration" value="SignModule_Conf1.xml" />
 </configuration>
</instance>
<!-- Settings module encryption -->
<configuration name="EncryptionConf">
    <!-- Specific encryption module -->
    <parameter name="class"
               value="eu.eidas.auth.engine.core.impl.EncryptionSW" />
    <!-- Settings specific module -->
    <parameter name="fileConfiguration" value="EncryptModule_Conf1.xml" />
</configuration>
<!-- Settings for the ExtensionProcessor module -->
<configuration name="ProtocolProcessorConf">
    <!-- Specific ExtensionProcessor module -->
    <parameter name="class"
               value="eu.eidas.sp.SpEidasProtocolProcessor"/>
    <parameter name="coreAttributeRegistryFile"
               value="saml-engine-eidas-attributes.xml" />
    <parameter name="additionalAttributeRegistryFile"
               value="saml-engine-additional-attributes.xml" />
    <parameter name="metadataFetcherClass"
               value="eu.eidas.sp.metadata.SPCachingMetadataFetcher"/>
</configuration>
```

All internal path elements are relative.

Each engine-instance needs four configuration steps:

1. SAML message configuration for eIDAS (`SamlEngineConf`).
2. Configuration of the Sign and Validation Module (`SignatureConf`).
    a. Specific Sign Module (`class`) and its configuration file (`fileConfiguration`).
    b. To create a new Sign and Validation module, it must implement the `SAMLEngineSignI` interface (`eu.eidas.auth.engine.core.SAMLEngineSignI`).
    c. The Sign module configuration file must be an xml file.
3. Configuration of the Encryption Module (`EncryptionConf`).
    a. Specific Encryption Module (`class`) and its configuration file (`fileConfiguration`).
    b. To create a new Encryption module, it must implement the `SAMLEngineEncryptionI` interface (`eu.eidas.auth.engine.core.SAMLEngineEncryptionI`).
    c. The Encryption module configuration file must be an xml file.
4. Configuration for the ExtensionProcessor Module:
    a. Attribute Registry configuration files. These are xml files.
    b. The files contain the eIDAS compliance attributes
    c. If needed, an additional attribute file is provided to allow declaration of sector specific attributes.

The following are the possible options to configure at the saml-engine behaviour when generating and validating attributes(`SamlEngine_Conf1.xml`):

```
<!--Types of consent obtained from the user for this authentication and data
transfer.Allow values: 'unspecified'.-->
      <entry key="consentAuthnRequest">unspecified</entry>

<!--Allow values: 'obtained', 'prior', 'curent-implicit', 'curent-explicit',
'unspecified'.-->
      <entry key="consentAuthnResponse">obtained</entry>

<!—URI representing the classification of the identifier. Allow values: 'entity'.--
>
      <entry key="formatEntity">entity</entry>

<!—The SOAP binding is only supported for direct communication between SP-MW and
VIdP-->
      <entry key="protocolBinding">HTTP-POST</entry>

<!--A friendly name for the attribute that can be displayed to a user -->
      <entry key="friendlyName">false</entry>

<!--Optional attributes-->
<entry key="eIDSectorShare">false</entry>
      <entry key="eIDCrossSectorShare">false</entry>
      <entry key="eIDCrossBorderShare">false</entry>
```



```
<!—Attributes with require option. Set this to true if you want to support optional
values-->
       <entry key="isRequired">true</entry>

<!--Subject cannot be confirmed on or after this second time(positive number)-->
       <entry key="timeNotOnOrAfter">300</entry>

<!--Validation IP of the response-->
       <entry key="ipAddrValidation">false</entry>

<!--One time use-->
<entry key="oneTimeUse">true</entry>
```

If the sign module is configured with `eu.eidas.auth.engine.core.impl.SignSW` it needs to configure the file (`SignModule_Conf1.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
	<comment>SWModule sign with JKS.</comment>
	<entry key="keyStorePath">/keystores/keycountry1.jks</entry>
	<entry key="keyStorePassword">passkey1</entry>
	<entry key="keyPassword">pass1</entry>
	<entry key="issuer">CN=eidas, C=ES</entry>
	<entry key="serialNumber">4BA0BD66</entry>
	<entry key="keyStoreType">JKS</entry>
	<!—Metadata signature configuration -->
	<entry key="metadata.keyStorePath">eidasKeyStore__METADATA.jks</entry>
	<entry key="metadata.keyStorePassword">keystorepassword</entry>
	<entry key="metadata.keyPassword">keypassword</entry>
	<entry key="metadata.issuer">issuerDN</entry>
	<entry key="metadata.serialNumber">serialNumber</entry>
	<entry key="metadata.keyStoreType">JKS</entry>
</properties>
```

If the encryption module is configured with `eu.eidas.auth.engine.core.impl.EncryptionSW` it needs to configure the file (`EncryptModule_Conf1.xml`):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
	<entry key="keyStorePath">/keystores/keycountry1.jks</entry>
	<entry key="keyStorePassword">passkey1</entry>
	<entry key="keyPassword">pass1</entry>
	<entry key="issuer">CN=eidas, C=ES</entry>
	<entry key="serialNumber">4BA0BD66</entry>
	<entry key="keyStoreType">JKS</entry>
<!-- Management of the encryption activation -->
<entry
key="encryptionActivation">c:\PGM\projects\configEidas\encryptionConf.xml</entry>
<entry key="responseToPointIssuer.BE">CN=local-demo-cert, OU=DIGIT, O=European
Comission, L=Brussels, ST=Belgium,C=BE</entry>
<entry key="responseToPointSerialNumber.BE">54C8F779</entry>
<!--  … other requesters to which the responses wil be encrypted …- - >

<!—private key to be used for decryption-->
<entry key="responseDecryptionIssuer">CN=local-demo-cert, OU=DIGIT, O=European
Comission, L=Brussels, ST=Belgium, C=BE</entry>
<entry key="serialNumber">54C8F779</entry>

</properties>
```

Finally, the SAML engine instantiation must be like this:

```
EIDASSAMLEngine engine = EIDASSAMLEngine.getInstance("CONF1");
```

If it is necessary, more instances could be created:

```
<instance name="CONF2">
--------
</instance>
EIDASSAMLEngine engine2 = EIDASSAMLEngine.getInstance("CONF2");
<instance name="CONF_SPAIN">
--------
</instance>
EIDASSAMLEngine engine = EIDASSAMLEngine.getInstance("CONF_SPAIN ");
```

10.4. Using eIDAS SAML Engine (public interfaces)

```
public byte[] generateEIDASAuthnRequest(String destination, String
serviceProvider, int qaal,  PersonalAttributeList personalAttributeList, String
assertionConsumerServiceURL
        throws EIDASSAMLEngineException;
```

destination: URI reference of the SAML Request processor this request is being sent to

serviceProvider: service provider name

qaal: authentication level

personalAttributeList: requested attribute list

assertionConsumerServiceURL

```
public byte[] generateEIDASAuthnResponse(String inResponseTo, String issuer,
StringassertConsumerURL, String ipAddress, PersonalAttributeList
personalAttributeList, boolean isHashing)
        throws EIDASSAMLEngineException;
```

inResponseTo: Request authentication token ID.

issuer: issuer of a SAML assertion or protocol message

assertConsumerURL

ipAddress: citizen IP

personalAttributeList

isHashing: attribute values are hashing

```
public byte[] generateEIDASAuthnResponseFail(String inResponseTo, String issuer,
String assertionConsumerServiceURL, String code, String subordinateCode, String
message, String ipAddress, boolean isHashing)
        throws EIDASSAMLEngineException;
```

issuer: issuer of a SAML assertion or protocol message

code: See *D5.8.1b Interface Specification 1.1, Chapter 6.2.5*

subordinateCode: See *D5.8.1b Interface Specification 1.1, Chapter 6.2.5*

message: See *D5.8.1b Interface Specification 1.1, Chapter 6.2.5*

```
public EIDASAuthnRequest validateEIDASAuthnRequest(byte[] tokenSaml)
        throws EIDASSAMLEngineException;
        tokenSaml: SAML token
```

```
public EIDASAuthnResponse validateEIDASAuthnResponse(byte[] tokenSaml, String
userIP)
        throws EIDASSAMLEngineException;
        tokenSaml: SAML token
        userIP: user IP
```

11. Support for additional attributes

By default, the EIDAS-SAMLEngine module only supports the attributes included in eIDAS minimum dataset (see the document '*eIDAS SAML Attribute Profile*' from the eIDAS Technical Sub-group, 22 June 2015).

11.1. Attribute registry

Attribute registry is responsible for holding and supplying information of types, value format and namespace for creating and validating requests and responses. The registry basically contains Attribute Definition objects built from custom XML files. There are two kinds of XML files, where one is holding eIDAS protocol-restricted attributes (`saml-engine-eidas-attributes.xml`) and the other supports additional attributes, sometimes referred as dynamic attributes (`saml-engine-additional-attributes_EidasNode.xml`). Each deployment artifact (IdP, SP and eIDAS-Node) has its own configuration files. Moreover every SAML Engine can have different configurations specified by `SamlEngine.xml` files.

The following is an example of code to introduce a new attribute to the XML configuration:

```
    <entry
key="19.NameUri">http://eidas.europa.eu/attributes/natural/NewSomething</entry>
    <entry key="19.FriendlyName">NEW_SOMETHING</entry>
    <entry key="19.PersonType">NaturalPerson</entry>
    <entry key="19.Required">false</entry>
    <entry
key="19.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entr
y>
    <entry key="19.XmlType.LocalPart">NewSomethingType</entry>
    <entry key="19.XmlType.NamespacePrefix">eidas-natural</entry>
```

For the `key` prefix number, take the last one and increment it. For eIDAS protocol the person type (natural or legal) must be specified and aligned with namespace.

11.2. Attribute registry validation and metadata support

In addition to the Attribute Registry XML files there is a hard-coded list of supported core attributes in `LegalPersonSpec` and `NaturalPersonSpec` collected together in the `EidasSpec` class, can be found in the saml-engine module. This is necessary to get a reference of attribute definitions to perform business rule-based validations on requests and replies.

The list of supported attributes is published in the Metadata.

The same configuration for additional attributes should be used for all modules (Service Provider, Connector, Proxy Service and Identity Provider).

12. ProtocolEngine Configuration

12.1. Obtaining a `ProtocolEngine` instance

In eIDAS-Node version 1.1, the `ProtocolEngine` (**`eu.eidas.auth.engine.ProtocolEngine`**) replaces the deprecated `SAMLEngine`.

The protocol engine is responsible for implementing the protocol between the eIDAS-Node Connector and the eIDAS-Node Proxy Service. The default protocol engine strictly implements the eIDAS specification.

However the protocol engine can be customised to implement protocols other than eIDAS. A `ProtocolEngine` instance is obtained from a `ProtocolEngineFactory` (**`eu.eidas.auth.engine.ProtocolEngineFactory`**).

There is a default `ProtocolEngineFactory`, **`eu.eidas.auth.engine.DefaultProtocolEngineFactory`** which uses the default configuration files.

You can obtain the protocol engine named " **`#MyEngineName#`** " by using the following statement:

```
ProtocolEngineI protocolEngine =
DefaultProtocolEngineFactory.getInstance().getProtocolEngine("#MyEngineName#");
```

You can also achieve the same result using a convenient method in `ProtocolEngineFactory` via the `getDefaultProtocolEngine` method:

```
ProtocolEngineI engine =
ProtocolEngineFactory.getDefaultProtocolEngine("#MyEngineName#");
```

12.2. Configuring protocol engines

Protocol engines are created from a *`ProtocolEngineConfiguration`* (**`eu.eidas.auth.engine.configuration.ProtocolEngineConfiguration`**).

`ProtocolEngineConfiguration` instances are obtained from a *`ProtocolEngineConfigurationFactory`* (**`eu.eidas.auth.engine.configuration.dom.ProtocolEngineConfigurationFactory`**).

There is a default *`ProtocolEngineConfigurationFactory`*: (**`eu.eidas.auth.engine.configuration.dom.DefaultProtocolEngineConfigurationFactory)`** which uses the default configuration files.

As you can imagine, the *`DefaultProtocolEngineConfigurationFactory`* is the factory used by the *`DefaultProtocolEngineFactory`* to configure default protocol engine instances.

You can create your own *`ProtocolEngineConfigurationFactory`* and use it to create your own *`ProtocolEngineFactory`* which would not rely on the default configuration files.

For example the following is a Spring configuration snippet to create a custom *ProtocolEngineConfigurationFactory* and the corresponding *ProtocolEngineFactory*:

```
<bean id="NodeSamlEngineConfigurationFactory"
class="eu.eidas.auth.engine.configuration.dom.ProtocolEngineConfigurationFactory">
    <constructor-arg value="SamlEngine.xml"/>
    <constructor-arg value="#{eidasConfigFilePath}"/>
    <constructor-arg value="#{eidasConfigRepository}"/>
</bean>

<bean id="NodeProtocolEngineFactory"
class="eu.eidas.auth.engine.ProtocolEngineFactory">
    <constructor-arg ref="NodeSamlEngineConfigurationFactory"/>
</bean>
```

12.2.1. The DefaultProtocolEngineConfigurationFactory

The default ProtocolEngineConfigurationFactory uses a configuration file called **SamlEngine.xml**.

The format of this file is as follows:

```
<instances>
     <instance name="MyEngineName">

        <configuration name="SamlEngineConf">
            <parameter name="fileConfiguration"
value="SamlEngine_MyEngineName.xml"/>
        </configuration>

        <configuration name="SignatureConf">
            <parameter name="class" value="eu.eidas.auth.engine.core.impl.SignSW"/>
            <parameter name="fileConfiguration"
value="SignModule_MyEngineName.xml"/>
        </configuration>
        <configuration name="EncryptionConf">
            <!-- Specific signature module -->
            <parameter name="class"
value="eu.eidas.auth.engine.core.impl.EncryptionSW"/>
            <!-- Settings specific module responseTo/FromPointAlias &
requestTo/FromPointAlias parameters will be added -->
            <parameter name="fileConfiguration"
value="EncryptModule_MyEngineName.xml"/>
        </configuration>

        <!-- Settings for the ExtensionProcessor module -->
        <configuration name="ProtocolProcessorConf">
            <!-- Specific ExtensionProcessor module -->
            <parameter name="class"
value="eu.eidas.auth.engine.core.eidas.EidasProtocolProcessor" />

            <parameter name="coreAttributeRegistryFile"
                       value="saml-engine-eidas-attributes-MyEngineName.xml" />
```

```
            <parameter name="additionalAttributeRegistryFile"
                       value="saml-engine-additional-attributes-MyEngineName.xml" />

        </configuration>

        <!-- Settings for the Clock module -->
        <configuration name="ClockConf">
            <!-- Specific Clock module -->
            <parameter name="class"
                       value="eu.eidas.auth.engine.SamlEngineSystemClock" />
        </configuration>

    </instance>

</instances>
```

The `SamlEngine.xml` file can be put on the `filesystem` or in the `classpath`. It is first looked for in the `classpath` but if not found, is loaded from the `filesystem`.

If the `SamlEngine.xml` file is available as a file URL (i.e. file://*somepath*), it is reloadable and will be reloaded as soon as the file is modified and its last-modified date attribute changes.

On the contrary, if the file is available from a jar, war or ear URL, it is not reloadable.

Therefore if you want the ProtocolEngine configuration to be reloadable at runtime, put it in a folder in the `classpath` outside of an archive (jar, war, ear).

The `SamlEngine.xml` file contains a sequence of instances. An instance represents one configuration of a ProtocolEngine. A given ProtocolEngine is obtained by its name (in the example "`MyEngineName`") which must be unique per configuration file.

An instance is mapped to a `ProcotolEngineConfiguration`.

The `ProtocolEngineConfiguration` is composed of:

- A unique name (e.g. `MyEngineName`)
- Core properties (which are configured via a configuration entry called `SamlEngineConf`)
- A Signature configuration entry (called `SignatureConf`)
- An (optional) encryption configuration entry (called `EncryptionConf`)
- A ProtocolProcessor configuration entry (called `ProtocolProcessorConf`)
- A Clock configuration entry (called `ClockConf`)

12.2.2. Core properties

Protocol engine core properties are configured using the following configuration entry:

```
<configuration name="SamlEngineConf">
    <parameter name="fileConfiguration" value="SamlEngine_MyEngineName.xml"/>
```

```
</configuration>
```

This entry is mapped to an implementation of the **`eu.eidas.auth.engine.core.SamlEngineCoreProperties`** interface.

Each core property can be configured directly inside the configuration entry (using parameters) or configured in an external file. The external file is referenced through a special parameter called "**`fileConfiguration`**". This file can be put on the `filesystem` or in the `classpath`. It is first looked for in the `classpath` but if not found, is loaded from the `filesystem`.

It is not necessary to use an external file, all core properties can be configured directly using parameters in the `SamlEngine.xml` file. If an external file is used, its format is a standard Java Properties file using either the .properties format or the .xml format.

If the Properties file is available as a file URL (i.e. file://*somepath*), it is reloadable and will be reloaded as soon as the file is modified and its last-modified date attribute changes.

On the contrary, if the file is available from a jar, war or ear URL, it is not reloadable.

Therefore if you want the core properties configuration to be reloadable at runtime, put it in a folder in the `classpath` outside of an archive (jar, war, ear).

The following is an example external file for core properties:

```
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>
   <comment>SAML constants for AuthnRequests and Responses.</comment>

   <!--
      Types of consent obtained from the user for this authentication and

      data transfer.
      Allow values: 'unspecified'.
   -->
   <entry key="consentAuthnRequest">unspecified</entry>

   <!--
   Allow values: 'obtained', 'prior', 'current-implicit', 'current-explicit',
'unspecified'.
   -->
   <entry key="consentAuthnResponse">obtained</entry>

   <!--URI representing the classification of the identifier
      Allow values: 'entity'.
   -->
   <entry key="formatEntity">entity</entry>

   <!--Only HTTP-POST binding is only supported for inter PEPS-->
   <!--The SOAP binding is only supported for direct communication between SP-MW
and VIdP-->
   <entry key="protocolBinding">HTTP-POST</entry>

   <!—eIDAS Node in the Service Provider's country-->
```

```
   <entry key="requester">http://eIDASNode.gov.xx</entry>

   <!-- eIDAS Node in the citizen's origin country-->
   <entry key="responder">http://eIDASNode.gov.xx</entry>

   <!--Subject cannot be confirmed on or after this seconds time (positive number)-
->
   <entry key="timeNotOnOrAfter">300</entry>

   <!--Validation IP of the response-->
   <entry key="ipAddrValidation">false</entry>
   <!--allow unencrypted responses-->
   <entry key="allowUnencryptedResponse">true</entry>

</properties>
```

The core property keys can be found in the **`eu.eidas.auth.engine.core.SAMLCore`** enum.

The most important core properties are:

- formatEntity
- ipAddrValidation
- oneTimeUse
- consentAuthnRequest
- timeNotOnOrAfter
- requester
- responder
- validateSignature
- consentAuthnResponse
- protocolBinding
- messageFormat.eidas

The list of all core property keys and values is available in section 4.1 — '*eIDAS-Nodes Configuration Files*'.

12.2.3. Signature Configuration

The following is an example of the signature configuration entry:

```
<configuration name="SignatureConf">
    <parameter name="class" value="eu.eidas.auth.engine.core.impl.SignSW"/>
    <parameter name="fileConfiguration" value="SignModule_MyEngineName.xml"/>
</configuration>
```

It contains a class parameter which must have as value the name of a class available in the `classpath` which implements the **`eu.eidas.auth.engine.core.ProtocolSignerI`** interface.

A base abstract class to implement this interface can be **`eu.eidas.auth.engine.core.impl.AbstractProtocolSigner`**.

Note: The given class should also implement the **`eu.eidas.auth.engine.metadata.MetadataSignerI`** interface to sign eIDAS metadata documents.

The configured implementing class must have a public constructor taking as single argument a **`java.util.Map of <String, String>`**.

The signature configuration must also contain signature properties.

The signature properties correspond to the **`eu.eidas.auth.engine.configuration.dom.SignatureConfiguration`** class and all signature property keys are defined in the **`eu.eidas.auth.engine.configuration.dom.SignatureKey`** enum.

Each signature property can be configured directly inside the configuration entry (using parameters) or configured in an external file. The external file is referenced through a special parameter called "`fileConfiguration`". This file can be put on the `filesystem` or in the `classpath`. It is first looked for in the `classpath` but if not found, is loaded from the `filesystem`.

It is not necessary to use an external file, all signature properties can be configured by using parameters directly in the **`SamlEngine.xml`** file. If an external file is used, its format is a standard Java Properties file using either the .properties format or the .xml format. If the Properties file is available as a file URL (i.e. file://*somepath*).

The following is an example of an external file for signature properties:

```
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
 <properties>
    <comment>SWModule sign with JCEKS.</comment>
    <entry key="check_certificate_validity_period">true</entry>
    <entry key="disallow_self_signed_certificate">true</entry>
    <entry key="keyStorePath">MyKeyStore.jceks</entry>
    <entry key="keyStorePassword">local-demo</entry>
    <entry key="keyPassword">local-demo</entry>
    <entry key="issuer">CN=local-demo-cert, OU=DIGIT, O=European Commission,
L=Brussels, ST=Belgium, C=BE</entry>
    <entry key="serialNumber">54c8f779</entry>
    <entry key="keyStoreType">JCEKS</entry>
</properties>
```

The following table shows the various encryption property keys:

Key	Description
`response.encryption.mandatory`	Specifies whether encryption must always be used.
`data.encryption.algorithm`	Specifies the data encryption algorithm (optional) (If not specified, the default value is http://www.w3.org/2009/xmlenc11#aes256-gcm).
`key.encryption.algorithm`	Specifies the key encryption algorithm (If not specified, the default value is http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p).

Key	Description
encryption.algorithm.whitelist	Specifies the white list of allowed data encryption algorithms (comma or semi-colon separated values) (optional) (If not specified, a default white list is used).
check_certificate_validity_period	Specifies whether certificate validity period is enforced or whether expired certificates can be used.
disallow_self_signed_certificate	Specifies whether self-signed certificates can be used
jcaProviderName	Specifies the name of the Java Cryptography Architecture (JCA) Security Provider to be used for the encryption (optional) (If none is specified, all the installed Security Providers are tried).
responseDecryptionIssuer	Specifies the issuer DN of the certificate used to perform decryptions. This is the certificate published in the metadata and advertised as the encryption certificate for this node.
serialNumber	Specifies the serialNumber of the certificate used to perform decryptions.
keyStorePath	Specifies the name of the keyStore to be loaded from the classpath or the path of the keyStore on the filesystem
keyStoreType	Specifies the type of the keyStore (optional) (If not specified, the default Java keyStore type is used).
keyStoreProvider	Specifies the provider of the keyStore (optional) (If not specified, all the installed security providers are tried).
keyStorePassword	Specifies the password of the keyStore.
keyPassword	Specifies the password of the private key used to perform decryptions.
responseToPointIssuer	Contains a valid country code : the serial number of a certificate to be used to encrypt responses to the given country when no metadata is used (not for eIDAS protocol).
responseToPointSerialNumber.	Contains a valid country code : the serial number of a certificate to be used to encrypt responses to the given country when no metadata is used (not for eIDAS protocol).
encryptionActivation:	Specifies the name of the encryption activation file to be loaded from the classpath or the path of the encryption activation file on the filesystem

12.2.4. The encryption activation file

Encryption can be activated country per country when the **response.encryption.mandatory** property is NOT set.

This is configured in the encryption activation file. The encryption activation file looks as follows:

```
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <entry key="EncryptTo.LU">true</entry>
    <entry key="DecryptFrom.LU">true</entry>

    <entry key="EncryptTo.FR">true</entry>
    <entry key="DecryptFrom.FR">true</entry>

    <entry key="EncryptTo.DE">true</entry>
```

```
    <entry key="DecryptFrom.DE">true</entry>

    <entry key="EncryptTo.BE">false</entry>
    <entry key="DecryptFrom.BE">false</entry>

</properties>
```

If the **response.encryption.mandatory** property is set to "true", this file is ignored.

The various encryption property keys are:

- **EncryptTo** + a valid country code — whether encryption is turned on to the given country.
- **DecryptFrom** + a valid country code — whether decryption is turned on from the given country

If the "**EncryptTo**. + a valid country code" property is not enabled, the Proxy Service does not encrypt the response sent to the Connector of the corresponding country.

The activation file properties can also be configured directly into the encryption file or protocol engine configuration file.

12.2.5. ProtocolProcessor configuration

The following is an example of the `ProtocolProcessor` configuration entry:

```
<!-- Settings for the ProtocolProcessor module -->
<configuration name="ProtocolProcessorConf">
    <!-- Specific ExtensionProcessor module -->
    <parameter name="class"
               value="eu.eidas.auth.engine.core.eidas.EidasProtocolProcessor" />

    <parameter name="coreAttributeRegistryFile"
               value="protocol-engine-eidas-attributes.xml" />

    <parameter name="additionalAttributeRegistryFile"
               value="protocol-engine-additional-attributes.xml" />

    <parameter name="metadataFetcherClass"
               value="eu.eidas.node.auth.metadata.SpringManagedMetadataFetcher"/>
</configuration>
```

It contains a **class** parameter which must have a value of the name of a class available in the `classpath` which implements the **eu.eidas.auth.engine.core.ProtocolProcessorI** interface.

An example of implementation for this interface can be **eu.eidas.auth.engine.core.eidas.EidasProtocolProcessor**.

The configured class implementing the `ProtocolProcessorI` interface is responsible for providing the actual protocol implementation.

eu.eidas.auth.engine.core.eidas.EidasProtocolProcessor provides the implementation of the eIDAS protocol.

Currently only SAML-based protocols are supported. The configured implementing class must have a **public** constructor with the following signature

```
public SomeProtocolProcessor(@Nullable AttributeRegistry specAttributeRegistry,
                             @Nullable AttributeRegistry
additionalAttributeRegistry,
                             @Nullable MetadataFetcherI metadataFetcher,
                             @Nullable MetadataSignerI metadataSigner) {...
```

The first constructor argument is the attribute registry of the protocol specification (for example, the eIDAS minimum data sets).

The second constructor argument is the attribute registry of additional attributes (aka sector-specific attributes).

The third constructor argument is the implementation of the `MetadataFetcherI` interface when metadata are used (mandatory for the eIDAS protocol).

The fourth constructor argument is the implementation of the MetadataSignerI interface when metadata signing is enabled (example for the eIDAS protocol).

The `ProtocolProcessor` configuration can also contain properties.

The `ProtocolProcessor` property keys are defined in the **eu.eidas.auth.engine.configuration.dom.ParameterKey** enum.

The various `ProtocolProcessor` property keys are:

- **coreAttributeRegistryFile** the name of the core attribute registry file to be loaded from the classpath or the path of the core attribute registry file on the filesystem (optional).
- **additionalAttributeRegistryFile** the name of the additional attribute registry file to be loaded from the classpath or the path of the additional attribute registry file on the filesystem (optional).
- **metadataFetcherClass** the name of a class available in the classpath which implements the eu.eidas.auth.engine.metadata.MetadataFetcherI interface (optional).

The core **AttributeRegistry** is the attribute registry of the protocol specification (for example, the eIDAS minimum data sets).

The additional **AttributeRegistry** is the attribute registry of additional attributes (aka sector-specific attributes).

When no core attribute registry is configured, the registry of the eIDAS specification is used (**eu.eidas.auth.engine.core.eidas.spec.EidasSpec.REGISTRY**).

When no additional registry is configured, there is simply no additional attribute at all.

12.2.6. The Attribute Registry

The attribute registry contains the definitions of all the supported attributes. The attribute registry is implemented in the class **`eu.eidas.auth.commons.attribute.AttributeRegistry`**. An attribute registry can be instantiated programmatically with the **`AttributeRegistry`** class or loaded from a file.

This file can be put on the filesystem or in the classpath. It is first looked after in the classpath but if not found, is loaded from the filesystem. Its format is a standard Java Properties file using either the .properties format or the .xml format.

If the Properties file is available as a file URL (i.e. file://somepath), it is reloadable and will be reloaded as soon as the file is modified and its last-modified date attribute changes. On the contrary if the file is available from a jar, war or ear URL, it is not reloadable.

Therefore if you want the attribute registry to be reloadable at runtime, put it in a folder in the classpath outside of an archive (jar, war, ear).

Example attribute registry complying with the eIDAS specification:

```
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <comment>eIDAS attributes</comment>
     <entry
key="1.NameUri">http://eidas.europa.eu/attributes/naturalperson/PersonIdentifier</e
ntry>
    <entry key="1.FriendlyName">PersonIdentifier</entry>
    <entry key="1.PersonType">NaturalPerson</entry>
    <entry key="1.Required">true</entry>
    <entry key="1.UniqueIdentifier">true</entry>
    <entry
key="1.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="1.XmlType.LocalPart">PersonIdentifierType</entry>
    <entry key="1.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="1.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralString
AttributeValueMarshaller</entry>


    <entry
key="2.NameUri">http://eidas.europa.eu/attributes/naturalperson/CurrentFamilyName</
entry>
    <entry key="2.FriendlyName">FamilyName</entry>
    <entry key="2.PersonType">NaturalPerson</entry>
    <entry key="2.Required">true</entry>
    <entry key="2.TransliterationMandatory">true</entry>
    <entry
key="2.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="2.XmlType.LocalPart">CurrentFamilyNameType</entry>
    <entry key="2.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="2.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.StringAttribu
teValueMarshaller</entry>
```



```
    <entry
key="3.NameUri">http://eidas.europa.eu/attributes/naturalperson/CurrentGivenName</e
ntry>
    <entry key="3.FriendlyName">FirstName</entry>
    <entry key="3.PersonType">NaturalPerson</entry>
    <entry key="3.Required">true</entry>
    <entry key="3.TransliterationMandatory">true</entry>
    <entry
key="3.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="3.XmlType.LocalPart">CurrentGivenNameType</entry>
    <entry key="3.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="3.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.StringAttribu
teValueMarshaller</entry>

    <entry
key="4.NameUri">http://eidas.europa.eu/attributes/naturalperson/DateOfBirth</entry>
    <entry key="4.FriendlyName">DateOfBirth</entry>
    <entry key="4.PersonType">NaturalPerson</entry>
    <entry key="4.Required">true</entry>
    <entry
key="4.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="4.XmlType.LocalPart">DateOfBirthType</entry>
    <entry key="4.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="4.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.DateTimeAttri
buteValueMarshaller</entry>

    <entry
key="5.NameUri">http://eidas.europa.eu/attributes/naturalperson/BirthName</entry>
    <entry key="5.FriendlyName">BirthName</entry>
    <entry key="5.PersonType">NaturalPerson</entry>
    <entry key="5.Required">false</entry>
    <entry key="5.TransliterationMandatory">true</entry>
    <entry
key="5.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="5.XmlType.LocalPart">BirthNameType</entry>
    <entry key="5.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="5.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.StringAttribu
teValueMarshaller</entry>

    <entry
key="6.NameUri">http://eidas.europa.eu/attributes/naturalperson/PlaceOfBirth</entry
>
    <entry key="6.FriendlyName">PlaceOfBirth</entry>
    <entry key="6.PersonType">NaturalPerson</entry>
    <entry key="6.Required">false</entry>
    <entry
key="6.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="6.XmlType.LocalPart">PlaceOfBirthType</entry>
    <entry key="6.XmlType.NamespacePrefix">eidas-natural</entry>
```

```
    <entry
key="6.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralString
AttributeValueMarshaller</entry>

    <entry
key="7.NameUri">http://eidas.europa.eu/attributes/naturalperson/CurrentAddress</ent
ry>
    <entry key="7.FriendlyName">CurrentAddress</entry>
    <entry key="7.PersonType">NaturalPerson</entry>
    <entry key="7.Required">false</entry>
    <entry
key="7.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="7.XmlType.LocalPart">CurrentAddressType</entry>
    <entry key="7.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="7.AttributeValueMarshaller">eu.eidas.auth.commons.protocol.eidas.impl.CurrentA
ddressAttributeValueMarshaller</entry>

    <entry
key="8.NameUri">http://eidas.europa.eu/attributes/naturalperson/Gender</entry>
    <entry key="8.FriendlyName">Gender</entry>
    <entry key="8.PersonType">NaturalPerson</entry>
    <entry key="8.Required">false</entry>
    <entry
key="8.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/naturalperson</entry
>
    <entry key="8.XmlType.LocalPart">GenderType</entry>
    <entry key="8.XmlType.NamespacePrefix">eidas-natural</entry>
    <entry
key="8.AttributeValueMarshaller">eu.eidas.auth.commons.protocol.eidas.impl.GenderAt
tributeValueMarshaller</entry>

    <entry
key="9.NameUri">http://eidas.europa.eu/attributes/legalperson/LegalPersonIdentifier
</entry>
    <entry key="9.FriendlyName">LegalPersonIdentifier</entry>
    <entry key="9.PersonType">LegalPerson</entry>
    <entry key="9.Required">true</entry>
    <entry key="9.UniqueIdentifier">true</entry>
    <entry
key="9.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="9.XmlType.LocalPart">LegalPersonIdentifierType</entry>
    <entry key="9.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="9.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralString
AttributeValueMarshaller</entry>

    <entry
key="10.NameUri">http://eidas.europa.eu/attributes/legalperson/LegalName</entry>
    <entry key="10.FriendlyName">LegalName</entry>
    <entry key="10.PersonType">LegalPerson</entry>
    <entry key="10.Required">true</entry>
    <entry key="10.TransliterationMandatory">true</entry>
    <entry
key="10.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="10.XmlType.LocalPart">LegalNameType</entry>
```

```
    <entry key="10.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="10.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.StringAttrib
uteValueMarshaller</entry>

    <entry
key="11.NameUri">http://eidas.europa.eu/attributes/legalperson/LegalAddress</entry>
    <entry key="11.FriendlyName">LegalAddress</entry>
    <entry key="11.PersonType">LegalPerson</entry>
    <entry key="11.Required">false</entry>
    <entry
key="11.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="11.XmlType.LocalPart">LegalPersonAddressType</entry>
    <entry key="11.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="11.AttributeValueMarshaller">eu.eidas.auth.commons.protocol.eidas.impl.LegalAd
dressAttributeValueMarshaller</entry>

    <entry
key="12.NameUri">http://eidas.europa.eu/attributes/legalperson/VATRegistration</ent
ry>
    <entry key="12.FriendlyName">VATRegistration</entry>
    <entry key="12.PersonType">LegalPerson</entry>
    <entry key="12.Required">false</entry>
    <entry
key="12.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="12.XmlType.LocalPart">VATRegistrationNumberType</entry>
    <entry key="12.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="12.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>

    <entry
key="13.NameUri">http://eidas.europa.eu/attributes/legalperson/TaxReference</entry>
    <entry key="13.FriendlyName">TaxReference</entry>
    <entry key="13.PersonType">LegalPerson</entry>
    <entry key="13.Required">false</entry>
    <entry
key="13.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="13.XmlType.LocalPart">TaxReferenceType</entry>
    <entry key="13.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="13.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>

    <entry key="14.NameUri">http://eidas.europa.eu/attributes/legalperson/D-2012-
17-EUIdentifier</entry>
    <entry key="14.FriendlyName">D-2012-17-EUIdentifier</entry>
    <entry key="14.PersonType">LegalPerson</entry>
    <entry key="14.Required">false</entry>
    <entry
key="14.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="14.XmlType.LocalPart">D-2012-17-EUIdentifierType</entry>
    <entry key="14.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="14.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>
```

```
    <entry
key="15.NameUri">http://eidas.europa.eu/attributes/legalperson/LEI</entry>
    <entry key="15.FriendlyName">LEI</entry>
    <entry key="15.PersonType">LegalPerson</entry>
    <entry key="15.Required">false</entry>
    <entry
key="15.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="15.XmlType.LocalPart">LEIType</entry>
    <entry key="15.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="15.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>

    <entry
key="16.NameUri">http://eidas.europa.eu/attributes/legalperson/EORI</entry>
    <entry key="16.FriendlyName">EORI</entry>
    <entry key="16.PersonType">LegalPerson</entry>
    <entry key="16.Required">false</entry>
    <entry
key="16.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="16.XmlType.LocalPart">EORIType</entry>
    <entry key="16.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="16.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>

    <entry
key="17.NameUri">http://eidas.europa.eu/attributes/legalperson/SEED</entry>
    <entry key="17.FriendlyName">SEED</entry>
    <entry key="17.PersonType">LegalPerson</entry>
    <entry key="17.Required">false</entry>
    <entry
key="17.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="17.XmlType.LocalPart">SEEDType</entry>
    <entry key="17.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="17.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>

    <entry
key="18.NameUri">http://eidas.europa.eu/attributes/legalperson/SIC</entry>
    <entry key="18.FriendlyName">SIC</entry>
    <entry key="18.PersonType">LegalPerson</entry>
    <entry key="18.Required">false</entry>
    <entry
key="18.XmlType.NamespaceUri">http://eidas.europa.eu/attributes/legalperson</entry>
    <entry key="18.XmlType.LocalPart">SICType</entry>
    <entry key="18.XmlType.NamespacePrefix">eidas-legal</entry>
    <entry
key="18.AttributeValueMarshaller">eu.eidas.auth.commons.attribute.impl.LiteralStrin
gAttributeValueMarshaller</entry>
</properties>
```

In practice such a file is not needed as it duplicates the **eu.eidas.auth.engine.core.eidas.spec.EidasSpec.REGISTRY.**

However the above example can be modified to tweak the eIDAS specification support.

The attribute registry file format

The attribute registry file is composed of attribute definitions. They represent the `eu.eidas.auth.commons.attribute.`***AttributeDefinition*** class. An attribute definition is composed of the following properties:

- **NameUri** : [mandatory]: the name URI of the attribute (full name and must be a valid URI)
- **FriendlyName** : [mandatory]: the friendly name of the attribute (short name)
- **PersonType** : [mandatory]: either NaturalPerson or LegalPerson .
- **Required** : [optional]: whether the attribute is required by the specification (and is part of the minimal data set which must be requested).
- **TransliterationMandatory** : [optional]: whether the attribute values must be transliterated if provided in non LatinScript variants. (This is mandatory for some eIDAS attributes).
- **UniqueIdentifier** : [optional]: whether the attribute is a unique identifier of the person (at least one unique identifier attribute must be present in authentication responses).
- **XmlType.NamespaceUri** : [mandatory]: the XML namespace URI for the attribute values, for example: http://www.w3.org/2001/XMLSchema for an XML Schema string
- **XmlType.LocalPart** : [mandatory]: the name of the XML type for the attributes values, for example: string for an XML Schema string
- **XmlType.NamespacePrefix** : [mandatory]: the name of the XML namespace prefix for the attributes values, for example: xs for an XML Schema string
- **AttributeValueMarshaller** : [mandatory]: the name of a class available in the classpath which implements the `eu.eidas.auth.commons.attribute.AttributeValueMarshaller` interface.

Each attribute definition in the property file is assigned a unique id followed by a dot '.' which allows the parser to associate properties to one given attribute definition.

The unique id can be any string not containing a dot '.'.

A convention can be to use numbers as unique ids as in the example above.

All properties used by the parser can be found in **eu.eidas.auth.commons.attribute.AttributeSetPropertiesConverter.**`Suffix` .

The **eu.eidas.auth.commons.attribute.AttributeValueMarshaller** interface is responsible for converting the string representation of an attribute value into a Java type and vice versa.

12.2.7. Clock configuration

The **`Clock`** configuration entry looks as follows:

```
<!-- Settings for the Clock module -->
<configuration name="ClockConf">
    <!-- Specific Clock module -->
    <parameter name="class"
               value="eu.eidas.auth.engine.SamlEngineSystemClock" />
</configuration>
```

It contains a *class* parameter which must have as value the name of a class available in the classpath which implements the **`eu.eidas.auth.engine.SamlEngineClock`** interface.

The configured implementing class must have a **`public`** *empty* constructor.

The clock interface is responsible for obtaining the system time.

12.2.8. Overriding the configuration with `eidas.xml`

In the eIDAS-Node, an override file is provided: `eidas.xml`. This file can redefine any configuration property accepted by the `ProtocolEngine` configuration files and the redefined value in **`eidas.xml`** prevails.

13. eIDAS-Node Security Headers

This section describes the HTTP response headers the eIDAS-Node can send in order to increase its security.

The web security model is based on the *same origin policy*. Code from https://EidasNode:8080/EidasNode/ should only have access to https://EidasNode:8080/EidasNode/ data, while an attacker from https://evil.eidasNode.com should certainly never be allowed access. Each origin is kept isolated from the rest of the web, giving developers a safe sandbox in which to build and play. While being a good concept in theory, in practice, attackers have found different ways to subvert the system.

Cross-site scripting (XSS) attacks for example bypass the same origin policy by tricking a site into delivering malicious code along with the intended content.

13.1. Available security headers

13.1.1. Content Security Policy

Content Security Policy (CSP) is a W3C standard that imposes restrictions on the origin of content. CSP prevents a wide range of attacks, including Cross-site scripting and other cross-site injections. It requires careful tuning and precise definition of the policy. If enabled, CSP has significant impact on the way browser renders pages (e.g., inline JavaScript is disabled by default and must be explicitly allowed in a policy).

Content Security Policy can significantly reduce the risk and impact of XSS attacks in modern browsers. Its mechanism is based principally on the definition of whitelists for the sources of content (applicable to a variety of resources: scripts, fonts, stylesheets, media, images, frames).

The implementation made in the eIDAS-Node applies a set of CSP policies on all HTTP responses returned by server.

```
content-security-policy    default-src 'none'; object-src 'self'; style-src 'self';
img-src 'self'; xhr-src 'self'; connect-src 'self';script-src 'self';script-nonce
903bcbf53880b2805eb1eafe3efd07df78857a59;report-uri
http://EidasNode:8080/EidasNode/cspReportHandler

x-content-security-policy  default-src 'none'; object-src 'self'; style-src 'self';
img-src 'self'; xhr-src 'self'; connect-src 'self';script-src 'self';script-nonce
903bcbf53880b2805eb1eafe3efd07df78857a59;report-uri
http://EidasNode:8080/EidasNode/cspReportHandler

X-WebKit-CSP default-src 'none'; object-src 'self'; style-src 'self'; img-src
'self'; xhr-src 'self'; connect-src 'self';script-src 'self';script-nonce
903bcbf53880b2805eb1eafe3efd07df78857a59;report-uri
http://EidasNode:8080/EidasNode/cspReportHandler
```

Figure 16: Sample CSP header in an HTTP response header of the eIDAS-Node

An action `cspReportHandler` has been defined for logging all the CSP violations.

```
WARN ContentSecurityPolicyReport [execute]:28 - Content security violation : {"csp-
report":{"document-
uri":"http://EidasNode:8080/EidasNode/ServiceProvider","referrer":"http://localhost
:8080/SP/IndexPage.action","blocked-uri":"self","violated-directive":"script-src
http://EidasNode:8080","source-
file":"http://EidasNode:8080/EidasNode/ServiceProvider","script-sample":"\n
document.getElementById(\"cspMessage...","line-number":51}}
```

Figure 17: Sample CSP violation reported by the cspReportHandler

If the browser is too old or if for another reason the CSP is not applied, a default error message will appear in the header of the application. This behaviour is disabled by default, because it is triggered by an enforced CSP violation, which is always displayed in server logs by the CSP report servlet.

Figure 18: Browser version is outdated

To enable this feature, `CSP.fallbackCheckMode` must be configured in eidas.xml.

For backward compatibility with older browser versions and CSP implementations, the headers `X-Content-Security-Policy` and `X-WebKit-CSP` are also added.

References :

http://www.html5rocks.com/en/tutorials/security/content-security-policy/

https://www.owasp.org/index.php/Content_Security_Policy

13.1.2. Mozilla original directive in CSP : xhr-src

The original Firefox implementation of CSP used the 'xhr-src' directive to restrict the origins to which an `XMLHttpRequest` object can connect. In the 1.0 spec, 'xhr-src' was replaced by 'connect-src' – which, in addition to XHR, also restricts where EventSource and WebSocket objects can connect.

```
content-security-policy    default-src 'none'; object-src 'self'; style-src 'self';
img-src 'self'; xhr-src 'self'; connect-src 'self';script-src 'self';script-nonce
903bcbf53880b2805eb1eafe3efd07df78857a59;report-uri
http://EidasNode:8080/EidasNode/cspReportHandler
```

Figure 19: Sample CSP header in an HTTP response header of the eIDAS-Node

13.1.3. X-XSS protection

This header enables the Cross-site scripting (XSS) filter built into most recent web browsers. It is usually enabled by default, meaning that the role of this header is to re-enable the filter for this particular website if it was disabled by the user. This header is supported in IE 8+, and in Chrome. The anti-XSS filter was added in Chrome 4.

```
X-XSS-Protection:   1; mode=block
```

Figure 20: Sample header in an HTTP response header of the eIDAS-Node

13.1.4. Strict-Transport-Security

HTTP Strict-Transport-Security (HSTS) instructs the browser to prefer secure connections to the server (HTTP over SSL/TLS) over insecure ones. This reduces the impact of bugs in web applications like leaking session data through cookies and external links, and defends against man-in-the-middle attacks. HSTS also disables the ability for users to ignore SSL certificate warnings.

For using this, the transport protocol needs to be https, and the following configuration has to be added in `web.xml`[1]:

```
<cookie-config>
  <secure>true</secure>
  <http-only>true</http-only>
</cookie-config>
```

Figure 21: Configuring HSTS in web.xml

```
Strict-Transport-Security:       max-age=600000; includeSubdomains
```

Figure 22: Sample HTTP response header of the eIDAS-Node

13.1.5. X-Frame-Options

Prevents the application from rendering in a frame or iframe, which in turns protects against key logging, clickjacking and similar attacks. Values: `deny` - no rendering within a frame, `sameorigin` - no rendering if origin mismatch, `allow-from`: `DOMAIN` - allow rendering if framed by frame loaded from DOMAIN.

---

[1] Note that the `web.xml` configuration of secure and http-only cookies has only been introduced with Servlet 3.0. For older platforms, this can be configured through proprietary mechanisms, such as application server specific web descriptors.

This option will prevent the eIDAS-Node from being framed in another application. The header value used by the eIDAS-Node is `sameorigin`.

```
X-Frame-Options: SAMEORIGIN
```

Figure 23: X-Frame-Options — Sample header in an HTTP eIDAS-Node response header

13.1.6. X-Content-Type-Options

The only defined value, `'nosniff'`, prevents Internet Explorer and Google Chrome from 'MIME-sniffing', that is, from trying to infer the actual MIME type of a response (which might be different from the declared content type) by inspecting its bytes. This also applies to Google Chrome, when downloading extensions. This header reduces the exposure to drive-by download attacks and sites serving user uploaded content that, by clever naming, could be treated as executable or dynamic HTML files.

```
X-Content-Type-Options: nosniff
```

Figure 24: X-Content-Type-Options — Sample header in an HTTP eIDAS-Node response header

13.1.7. Cache control – pragma expiration

Browsers can store information for purposes of caching and history. Caching is used to improve performance, so that previously displayed information does not need to be downloaded again. History mechanisms are used for user convenience, so the user can see exactly what they saw at the time when the resource was retrieved. If sensitive information is displayed to the user (e.g. their address, credit card details, Social Security Number, or username), then this information could be stored for purposes of caching or history, and therefore retrievable through examining the browser's cache or by simply pressing the browser's **Back** button.

The following is an example of a good header.

```
HTTP/1.1:
Cache-Control: no-cache
HTTP/1.0:
Pragma: no-cache
Expires: <past date or illegal value (e.g., 0)>
```

13.2. Development guidelines

The core issue exploited by XSS attacks is the browser’s inability to distinguish between scripts that are intended to be part of your application, and scripts that have been maliciously injected by a third party. Instead of making browsers trust blindly *everything* that a server delivers, CSP defines the `Content-Security-Policy` HTTP header that allows the application owner to create a whitelist of sources for trusted content, and instructs the browser to only execute or render resources from those sources.

Even if an attacker can find a hole through which to inject scripts, the origin of the script will not match the whitelist, and therefore will not be executed. In the eIDAS-Node, we specify `'self'` as the only valid source of scripts. With this policy defined, the browser will simply throw an error instead of loading scripts from any other source. If a clever attacker does manage to inject code into the site , he will run headlong into an error message, rather than having the success he was expecting.

The policy applies to a wide variety of resources:

- **script-src** limits the origins from which you can load scripts;
- **connect-src** limits the origins to which you can connect (via XHR, WebSockets, and EventSource);
- **font-src** specifies the origins that can serve web fonts. Google's Web Fonts could be enabled via font-src https://themes.googleusercontent.com;
- **frame-src** lists the origins that can be embedded as frames. For example: frame-src https://youtube.com would enable embedding YouTube videos, but no other origins;
- **img-src** defines the origins from which images can be loaded;
- **media-src** restricts the origins allowed to deliver video and audio;
- **object-src** allows control over Flash and other plugins;
- **style-src** is the counterpart of script-src for stylesheets.

In the eIDAS-Node, we use the **`'self'`** source that matches only the current origin and not its subdomains.

As a consequence, inline JavaScript and CSS are not allowed, and neither are **`eval`** expressions.

To give an example of the impact, the following inline JavaScript

```
<body onload="document.redirectForm.submit();">
```

will need to be replaced by a JavaScript file as shown below.

```
<script src="js/redirectOnload.js"></script>
```

This file would contain the following.

```
function submitRedirectFormAction() {
document.getElementsByName('redirectForm')[0].submit();
}
window.addEventListener('load', submitRedirectFormAction);
```

14. Clustering Environment

14.1. Introduction

This section describes the technologies and configurations used for testing the eIDAS-Node in cluster mode. The choice of technologies is proposed for testing purpose.

14.1.1. Load balancer

The configuration adopted is the following:

- One load balancer composed of two Tomcat 7 (version 7.0.55) servers including the eIDAS-Node;
- One Apache Http server to isolate SP/IDP request.

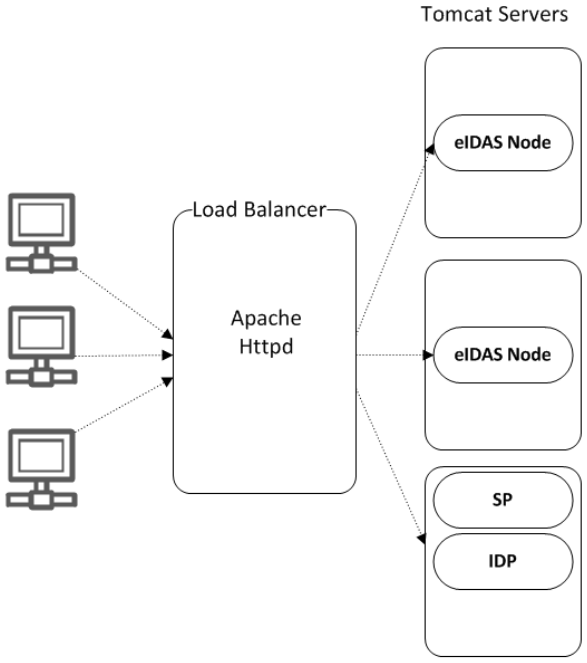


Figure 25: Clustering environment — Load balancer

The solution is to add one server in-front of all tomcat clusters to accept all the requests and distribute to the cluster. So this server acts as a **load balancer**.

There are several servers available with load balancing capability. Here we are going to use **Apache httpd** web server as a load balancer. With **mod_jk** module.

If one of the Tomcat instances fails then the load balancer dynamically reacts by ceasing to forward requests to that failed Tomcat instances. Other Tomcat instances continue as normal.

If the failed Tomcat is recovered from the failed state to normal state the load balancer will include it in the cluster to receive requests.

14.1.2. Load Balancer with Hazelcast

Hazelcast gives **High availability and full fail-over capability** to our clustering environment.

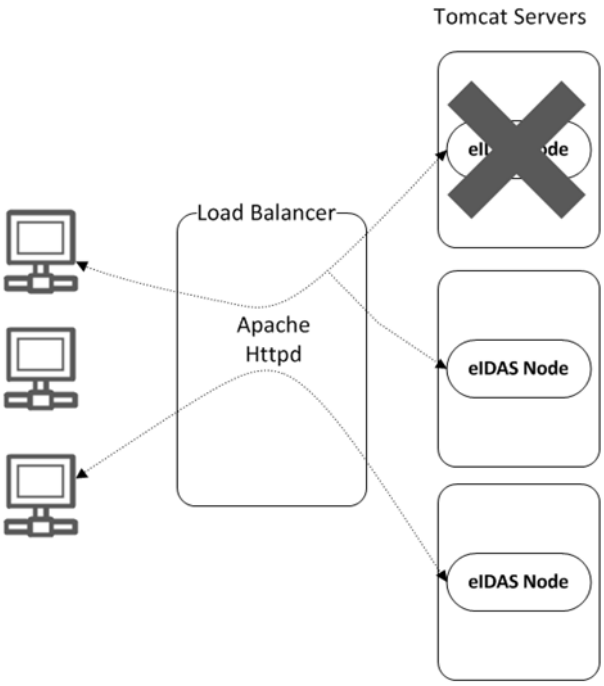


Figure 26: Clustering environment — Load Balancer with Hazelcast

14.2. Configuring Tomcat

14.2.1. Setting AJP ports

Traffic is passed between Apache and Tomcat(s) uses the binary AJP 1.3 protocol

Application Server	Http port	AJP port	Requests
Tomcat 7 – instance 1	Tomcat 1 port	8209	Connector,Proxy Service
Tomcat 7 – instance 2	Tomcat 2 port	8309	Connector,Proxy Service
Tomcat 7 - instance 3	Tomcat 3 port	8409	SP, IDP

14.2.2. Apache HTTPD

In this section we will use **Apache httpd** web server as a Load Balancer.

To provide the load balancing capability to Apache httpd server we need to include the module **mod_jk**.

14.2.2.1. Install and configure mod_jk

The **mod_jk** module is downloaded from http://www.apache.org/dist/tomcat/tomcat-connectors/jk/binaries/.

mod_jk is the Apache HTTPD module that will be used to provide our cluster with its load balancing and proxy capabilities, by default it uses the 'round robin' algorithm to distribute the requests.

It uses the AJP protocol to facilitate fast communication between Tomcat servers and the Apache Web Server that will receive the client requests

Configuration consists of adding a few lines to the main Apache HTTPD configuration file, `httpd.conf`.

```
JkMount  /status  stat
JkMount  /EidasNode/*  balancer
JkMount  /SP/* tomcat3
JkMount  /IdP/* tomcat3
```

14.2.2.2. Configure the cluster workers

'Workers' is a blanket term used within **mod_jk** to refer to both real Tomcat servers that will process requests, and virtual servers included in the module to handle load balancing and monitoring.

File: `workers.properties`

By default, mod_jk includes three additional load balancing algorithms, some of which are more appropriate for certain situations, and can be configured with the 'method' directive:

```
worker.list=balancer,stat,tomcat3
worker.tomcat1.type=ajp13
worker.tomcat1.port=8209
worker.tomcat1.host=localhost
worker.tomcat2.type=ajp13
worker.tomcat2.port=8309
worker.tomcat2.host=localhost
worker.tomcat3.type=ajp13
worker.tomcat3.port=8409
worker.tomcat3.host=localhost
worker.balancer.type=lb
worker.balancer.balance_workers=tomcat1,tomcat2
```

14.3. Set up Hazelcast

To replicate required information between cluster members, all nodes need to be configured with Hazelcast. Please refer to section 4.1.4.5 and Appendix D for information on how to implement the required configuration.

14.4. Check your installation

Open the Apache status page: http://localhost/status and check that each node is up and running.

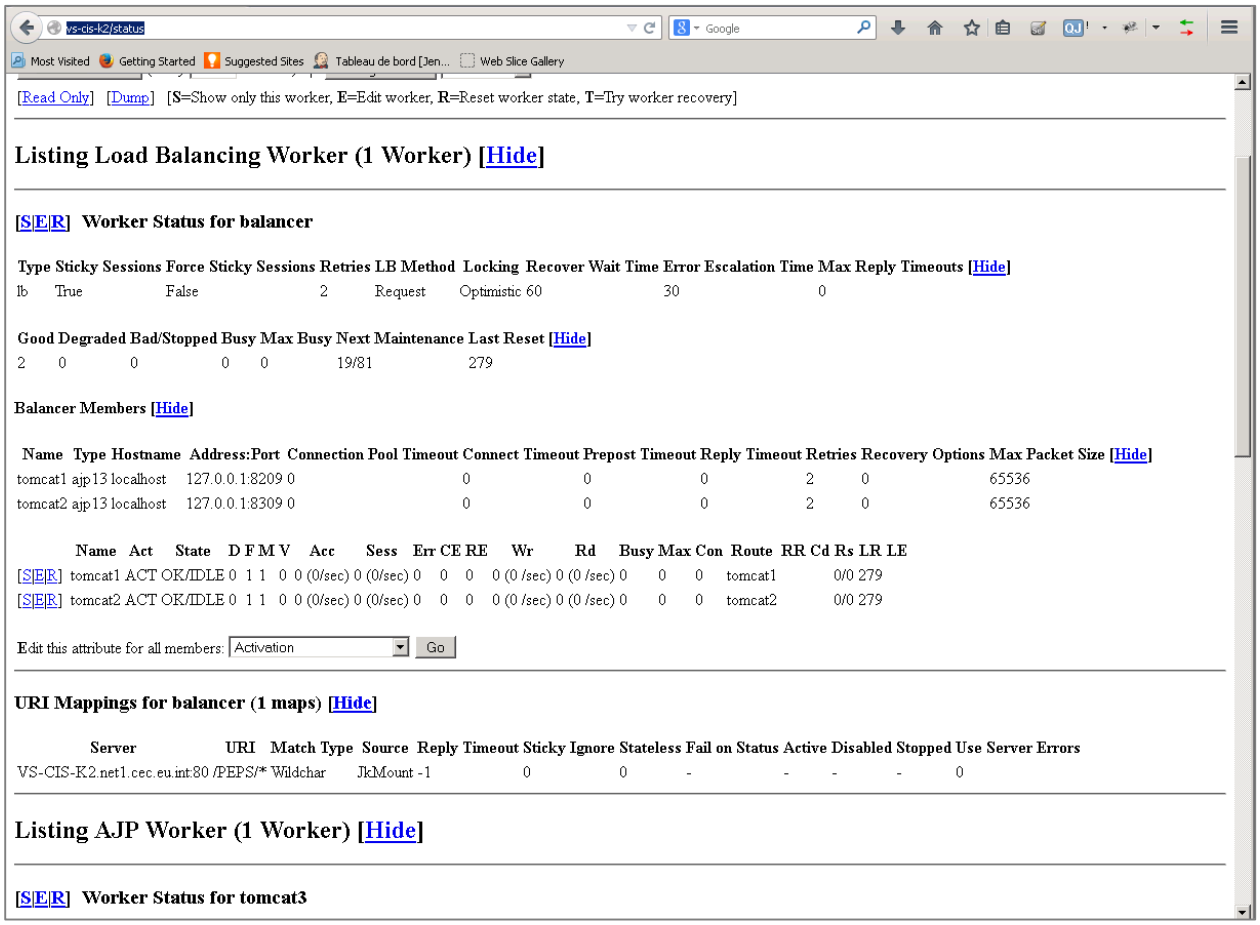
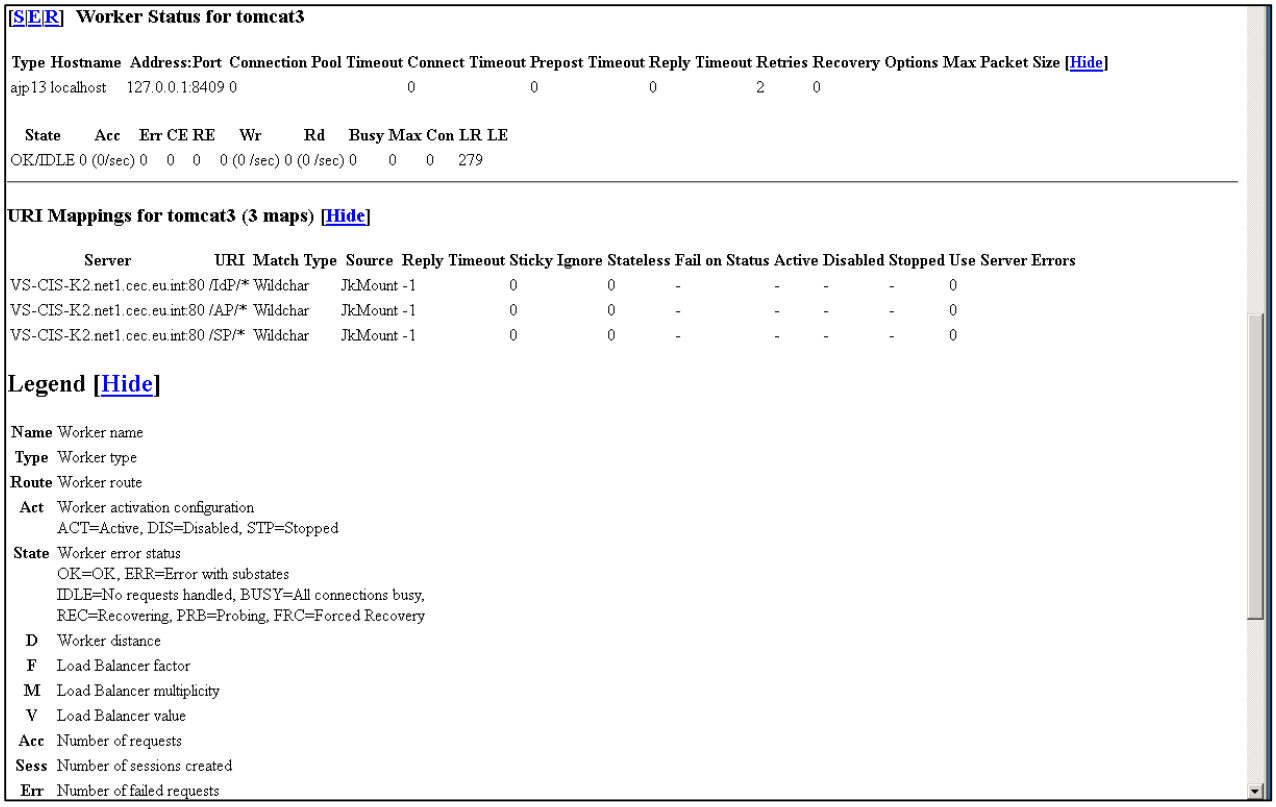


Figure 27: Apache status page



[S|E|R] **Worker Status for tomcat3**

Type	Hostname	Address:Port	Connection Pool Timeout	Connect Timeout	Prepost Timeout	Reply Timeout	Retries	Recovery Options	Max Packet Size [Hide]
ajp13	localhost	127.0.0.1:8409	0	0	0	0	2	0	

State	Acc	Err	CE	RE	Wr	Rd	Busy	Max	Con	LR	LE
OK/IDLE	0 (0/sec)	0	0	0	0 (0 /sec)	0 (0 /sec)	0	0	0	279	

URI Mappings for tomcat3 (3 maps) [Hide]

Server	URI	Match Type	Source	Reply Timeout	Sticky	Ignore Stateless	Fail on Status	Active	Disabled	Stopped	Use Server Errors
VS-CIS-K2.net1.cec.eu.int:80	/IdP/*	Wildchar	JkMount -1	0	0	-	-	-	-	0	
VS-CIS-K2.net1.cec.eu.int:80	/AP/*	Wildchar	JkMount -1	0	0	-	-	-	-	0	
VS-CIS-K2.net1.cec.eu.int:80	/SP/*	Wildchar	JkMount -1	0	0	-	-	-	-	0	

Legend [Hide]

Name	Worker name
Type	Worker type
Route	Worker route
Act	Worker activation configuration ACT=Active, DIS=Disabled, STP=Stopped
State	Worker error status OK=OK, ERR=Error with substates IDLE=No requests handled, BUSY=All connections busy, REC=Recovering, PRB=Probing, FRC=Forced Recovery
D	Worker distance
F	Load Balancer factor
M	Load Balancer multiplicity
V	Load Balancer value
Acc	Number of requests
Sess	Number of sessions created
Err	Number of failed requests

Figure 28: Apache status page (continued)

15. eIDAS-Node Error and Event Logging

15.1. Introduction

This section provides information on the eID implementation of error and event logging as a building block for generating an audit trail of activity on the eIDAS Network. It describes the files that are generated, the file format, the components that are monitored and the events that are recorded. It is also presents points that should be considered when planning, implementing and operating an auditing environment.

15.2. What is an audit trail?

A log is a record of the events occurring within an organisation's systems and networks. Logs are composed of log entries; each entry contains information related to a specific event that has occurred within a system or network.

An audit trail consists of a number of log records providing documentary evidence of the sequence of activities that have affected at any time a specific operation, procedure or event.

In a stand-alone system all the security relevant activity in individual components can be isolated, thus allowing us to maintain a time-ordered list of all actions and events (e.g. audit trail) that happened in the system and enabling us to audit the system by simply following the information trail in a single place, obeying to a defined notation and in a specific timeline.

The number, volume and variety of different systems operating in an eIDAS-Network's distributed architecture has created the need for security log management and safeguards to help protect the confidentiality, integrity, and availability of service. With this in mind an audit trail must meet two very important requirements:

- The possibility of reconstructing an entire transaction by linking all related request/response identifiers throughout the complete path from Service Provider to Identity Provider and back; starting from the end.
- The association between an incoming request and (all related) outgoing request(s), as well as the corresponding replies

In order to meet these requirements the eIDAS-Node implementation is required to record and retain audit-logging information sufficient to answer the following questions:

- Who performed the logging (the main component or an external module/party)?
- What activity was performed? (e.g. Input, Process, Output)
- Who or what performed the activity, including where or on what system the activity was performed from (subject)?
- What the activity was performed on (object)?
- When was the activity performed?
- What tool(s) was used to perform the activity?
- What was the status (such as success vs. failure), outcome, or result of the activity

15.3. Forensics and diagnostics

Analysis of the audit trail will help to identify use and abuse of the eIDAS Network, including the following conditions:

- Sequencing failure
- Excessive use
- Data changes
- Fraud and other criminal activities
- Suspicious, unacceptable or unexpected behavior
- Modifications to configuration
- Application code file and/or memory changes input validation failures e.g. protocol violations, unacceptable encodings, invalid parameter names and values
- Output validation failures e.g. database record set mismatch, invalid data encoding
- Authentication successes and failures
- Authorisation (access control) failures
- Session management failures e.g. cookie session identification value modification
- Application errors and system events e.g. syntax and runtime errors, connectivity problems, performance issues, third party service error messages, file system errors, file upload virus detection, configuration changes
- Application and related systems start-ups and shut-downs, and logging initialisation (starting, stopping or pausing)
- Use of higher-risk functionality e.g. network connections, addition or deletion of users, changes to privileges, assigning users to tokens, adding or deleting tokens, use of systems administrative privileges, access by application administrators, all actions by users with administrative privileges, access to payment cardholder data, use of data encrypting keys, key changes, creation and deletion of system-level objects, data import and export including screen-based reports, submission of user-generated content - especially file uploads
- Legal and other opt-ins e.g. permissions for mobile phone capabilities, terms of use, terms & conditions, personal data usage consent, permission to receive marketing communications

15.4. Standards, frameworks and best practice

Standards and frameworks like COBIT (Control Objectives for Information and Related Technology) by the Technology IT Governance Institute and the Information Systems Audit and Control Association (ISACA) and ITIL (Information Technology Infrastructure Library) provide information on IT governance and auditing. However there is little regulatory guidance on the specific controls, the standard of due care means doing at least what one's peers are doing as well as following security best practices frameworks as proposed by COBIT, BITS, COSO and standards including ISO 17799 and NIST SP 800-53 and ISF Standard of Good Practice, ISO / ISO 27002.

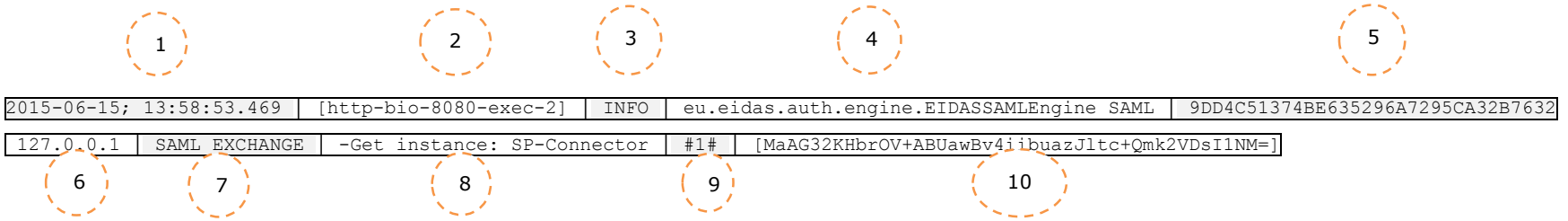
15.5. Log generation

The following table shows the information attributes that are contained in a log record.

Table 37: Log record attributes

Name	See Figure 29	Description
Date time	1	The date time of the of the logging event
Thread	2	Outputs the name of the thread that generated the logging event.
Level	3	Outputs the level of the logging event. • **ERROR**: some type of failure has occurred, users are probably being affected (or will soon be) and the fix probably requires human intervention. • **WARN**: an unexpected technical or business event occurred, users may be affected, but probably no immediate human intervention is required. Support personnel will want to review these issues a soon as possible to understand what the impact is. Basically any issue that needs to be tracked but may not require immediate intervention. • **INFO**: things we want to see at high volume in case we need to forensically analyse an issue. Typical business exceptions can go here (e.g. login failed due to bad credentials). • **DEBUG**: used for entry/exit of most non-trivial methods and detailed information on the flow through the system. This will be written to logs only. • **TRACE**: this would be for extremely detailed and potentially high volume logs that you do not typically want enabled even during normal development.
ClassName	4	Outputs location information of the caller which generated the logging event.
sessionId	5	If available the session Id.
IpAddress	6	If available the Remote IP address.
Event	7	If available, the type of logging event.
Message	8	The content of the message
EventNumber	9	Sequential increasing number of the event
Hash	10	A hash generated of the line log content guaranteeing integrity

The following is an example of log content showing three log records.




```
2015-06-15; 13:58:57.825 [http-bio-8080-exec-2] DEBUG eu.eidas.auth.engine.EIDASSAMLEngine -9DD4C51374BE635296A7295CA32B7632 -
127.0.0.1 SAML_EXCHANGE -null #2# [/d5F9UqFZy3OY37PA500TGpepPczdUzbJigiZxGIa2E=]


2015-06-15; 13:59:07.212 [http-bio-8080-exec-2] INFO  eu.eidas.node.auth.connector.AUCONNETORSAML -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 SAML_EXCHANGE -Connector - Processing SAML Request with ID
_a43526c05fc8168879b5789ba0d62744 #3# [AeRp3ofGkl3gcIEH4QypViREHPB9Kw+hFFpt1DdUKGc=]
```

Figure 29: Example of log content

Notes: The following shows the attribute delimiters and their meanings:

is used as a field delimiter

[ ] delimits optional components

Important: Additional attributes may be added in future phases of this project.

Figure 30 and Figure 31 below show examples of log records on a Service Provider and on an eIDAS-Node.

```
1#13Feb200918:22:59#Auth#POLITO#201.202.203.204#IT-eIDASNode#193.194.195.196#12345#
654ACEFD#
……
25#13Feb200918:23:21#Auth#IT-
eIDASNode#193.194.195.196#POLITO#201.202.203.204#34567#
ABE5737D#POLITO#12345
……
```

Figure 30: Log records on a Service Provider

```
2015-06-15; 13:58:53.469 [http-bio-8080-exec-2] INFO
eu.eidas.auth.engine.EIDASSAMLEngine SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Get instance: SP-Connector  #1#
[MaAG32KHbrOV+ABUawBv4iibuazJltc+Qmk2VDsI1NM=]
2015-06-15; 13:59:07.212 [http-bio-8080-exec-2] INFO
eu.eidas.node.auth.connector.AUCONNECTORSAML SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Connector - Processing SAML Request
with ID _a43526c05fc8168879b5789ba0d62744 #3#
[AeRp3ofGkl3gcIEH4QypViREHPB9Kw+hFFpt1DdUKGc=]
...
2015-06-15; 13:59:07.393 [http-bio-8080-exec-2] INFO
eu.eidas.auth.engine.EIDASSAMLEngine SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Get instance: Connector-Service  #15#
[lJCReY/HA7Kcc9HyKuJww2qFRvjSxkQfmw/1sBGnxH8=]
2015-06-15; 13:59:07.529 [http-bio-8080-exec-2] INFO
eu.eidas.node.auth.connector.AUCONNECTORSAML SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Connector - Processing SAML Request
with ID _a43526c05fc8168879b5789ba0d62744 #16#
[zsSBlcAwF8Lg1Ig7u2S//OtoklvnnLZnAPcWX6Quci8=]
...
2015-06-15; 13:59:08.445 [http-bio-8080-exec-2] INFO
eu.eidas.auth.engine.EIDASSAMLEngine SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Get instance: Service  #29#
[/K59pCfDIfHDTBiQKc1n5VLKBtdOprWsEumnndh8TIQ=]
2015-06-15; 13:59:08.697 [http-bio-8080-exec-2] INFO
eu.eidas.node.auth.service.AUSERVICESAML SAML_EXCHANGE -
9DD4C51374BE635296A7295CA32B7632 -127.0.0.1 -Service - Processing SAML Request with
ID _d41ae1dc063c7e4a87f17e57fdbc1329 #31#
[rkLbukPhl+5ia+wzzJyVY2DHHfz2JlXPvpT1CmHRZgw=]
```

Figure 31: Log records on IT-eIDAS-Node

15.6. Logging configuration settings

You should ensure that other, guidelines, and procedures that have some relationship to logging incorporate and support these log management requirements and recommendations, and also comply with functional and operational requirements. The following table provides examples of the types of logging configuration settings to be specified in policies which are MS specific.

Table 38: Logging configuration settings

Category	Low Impact Systems	Moderate Impact Systems	High Impact Systems
How long to retain log data	1 to 2 weeks	1 to 3 months	3 to 12 months
How often to rotate logs	Optional (if performed, at least every week or every 25 MB)	Every 6 to 24 hours, or every 2 to 5 MB	Every 15 to 60 minutes, or every 0.5 to 1.0 MB
If the organisation requires the system to transfer log data to the log management infrastructure, how frequently that should be done	Every 3 to 24 hours	Every 15 to 60 minutes	At least every 5 minutes
How often log data needs to be analysed locally (through automated or manual means)	Every 1 to 7 days	Every 12 to 24 hours	At least 6 times a day
Whether log file integrity checking needs to be performed for rotated logs	Optional	Yes	Yes
Whether rotated logs need to be encrypted	Optional	Optional	Yes
Whether log data transfers to the log management infrastructure need to be encrypted or performed on a separate logging network	Optional	Yes is feasible	Yes

15.7. Log files

Each eIDAS Node is configured to generate four different locally stored log files for tracing activities:

- **The Application System log:** (`eIDASNodeSystem`) contains all the events occurring at the application level (e.g. server start/stop, change of configuration).
- **The Application Security log**: (`eIDASNodeSecurity`) contains all the security related events.
- **The Message Exchange log:** (`eIDASNodeSAMLExchange`) contains the details about exchanged messages.
- **The Application Detailed log**: (`eIDASNodeDetail`) contains the more detailed information at destination to the technical team for forensics, investigation and debugging purposes.

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.

If modification of the environment variable is not possible, the value of this variable could also be assigned by adding the following line in the `logback.xml` file
`<property name="LOG_HOME" value ="<myDirectoryName>" />`

The following table shows a matrix of which events are logged, in which log file and which response would be triggered.

Table 39: Event log matrix

		Record in log file						Response type			
Event	**System**	**Detailed**	**Security**	**Message Exchg**	**Console**	**Appl. server log**	**Threshold level (see Table 37)**	**Immediate**	**Routine**	**Forensics - historic**	**Comment**
SAML request processing [EVT-1]		√		√	√		Info		√		Normal processing of the SAML messages
SAML Response processing [EVT-2]		√		√	√		Info		√		Normal processing of the SAML messages
OpenSAML Error [EVT-3]		√		√	√		Error	√			Error occurs in the OpenSAML product library
Configuration change [EVT-4]	√				√				√	√	Configuration change via management console
Configuration reload [EVT-5]	√				√				√	√	
Application startup [EVT-6]	√					√			√	√	Start of application filters
Application shutdown [EVT-4]	√					√			√	√	Start of application filters
Console alerts [EVT-5]	√	√			√			√		√	

		Record in log file						Response type			
Event	**System**	**Detailed**	**Security**	**Message Exchg**	**Console**	**Appl. server log**	**Threshold level (see Table 37)**	**Immediate**	**Routine**	**Forensics - historic**	**Comment**
Exceptions Application errors and system events e.g. syntax and runtime errors, connectivity problems, performance issues, file system errors [EVT-6]		√			√			√			
Unauthorised access attempt to management console [EVT-7]		√	√		√			√			
Input validation failures [EVT-8]		√	√		√				√		Could hide an attack
Output validation failures [EVT-9]		√	√		√				√		Could hide an attack
Session management failures e.g. cookie session identification value modification [EVT-10]		√	√		√			√			Could hide an attack
Content security violation [EVT-11]		√	√		√			√			Could hide an attack

		Record in log file						Response type			
Event	**System**	**Detailed**	**Security**	**Message Exchg**	**Console**	**Appl. server log**	**Threshold level (see Table 37)**	**Immediate**	**Routine**	**Forensics - historic**	**Comment**
Sequencing failure (like SAML Response without being preceded by SAML Request) [EVT-12]		√	√		√			√			Could hide an attack
Excessive use (like SAML request limitation exceeded) [EVT-13]		√	√		√			√			Could hide an attack
Certificate not valid [EVT-13]		√	√		√			√			Could hide an attack
Signature not valid [EVT-14]		√	√		√			√			Could hide an attack

15.8. Event detailed error codes and associated actions

For full information on eIDAS-Node error codes and associated actions, refer to the *eIDAS-Node Error Codes* document.

15.9. Operational considerations

- Audit facilities should be designed with their specific use(s) in mind, not simply adapted from existing system logs.
- Member States should create and maintain an audit log management structure, including policy, procedures, rules and tools.
- Audit log files should be periodically saved and moved to another storage space. The periodicity may be influenced by the classification of the data, but should be defined either according to a fixed size (e.g. every time the log reaches 1Mb) and / or according to a fixed period of time (e.g. per day, per week).
- Attacks on systems should be prevented by the use of technical preventive controls. These controls should be preferred above procedural preventive controls if possible. Both measures should be used in conjunction with examination of the audit data.
- Member States should consider the use of a Security Event management system due to the large volume of security events.
- Member States should not solely rely on unauthorised events or breaches of policy, but should additionally consider proactive monitoring.
- Member States should consider the use of Security Information and Event Management (SIEM) technology to help the automation and collection of auditable events.
- Audit records should contain a time stamp. Ensure that each system's clock is synchronised to a common time source so that its timestamp will match those generated by other systems.
- The privacy of personal data must be protected.
- Measures should be implemented in order that deleting or modifying logs of own activities should be detected and alerted immediately.
- Administrators should not be able to erase or de-activate logs of activities.
- Audit events should be protected from modification by using digital signatures to sign audit records.
- Audit events should be archived to 'write-once 'media to protect the archive from modification or deletion.
- Where audit records are subject to cryptographic authentication the input data to the authentication computation should include an accurate timestamp.
- Archived audit log files should be protected by appropriate logical and physical security mechanisms.
- The capture of customer sensitive data in audit logs should be avoided. If necessary sensitive data should be anonymised, tokenised or encrypted to avoid data breaches.
- Where audit data are encrypted, the appropriate decryption software and keys must be available and properly maintained, under appropriate access control, for the whole of the lifetime of the encrypted data. Proper maintenance should include periodic testing and adequate back-up to cater for loss of stored encryption keys.
- Audit records should be created in a simple standard format.

- Where audit data are compressed, the appropriate decompression software must be available and properly maintained for the whole of the lifetime of the compressed data. Proper maintenance should include periodic testing.
- The audit trails must be stored in a long-term support and a tamper evident format, such that illicit addition, modification or deletion of any audit trail can be detected.

15.9.1. Retention period

- MS should ensure that audit retention is part of the organisation's overall retention policy.
- Where the execution of sensitive security-related actions cannot be made subject to dual control, then that execution should be monitored in a timely fashion.
- MS should determine which data is classified as audit data and should protect and preserve this data appropriately.
- Personal notes should be kept by incident investigators throughout the whole course of an investigation and those notes should themselves be protected and preserved in the same manner as audit data.
- Audit records used during an investigation must be preserved at least until the end of the investigation and for any subsequent prosecution irrespective of their normal retention period.
- Follow the organisation's incident response policy to investigate an audit log incident.
- System configuration information should be modified, if necessary, to prevent an event from overwhelming the system.

15.10. References

[1] ISO/IEC 27002 - Information technology -- Security techniques -- Code of practice for information security management, section 10.10, 2005 (www.iso.org)

[2] BSI PD008: Legal Admissibility and Evidential Weight of Information Stored Electronically, British Standards Institution, 1999

[3] COBIT (Control Objectives for Information and related Technology) from Information Systems Audit and Control Association (http://www.isaca.org/cobit.htm)

[4] ICT-PSP/2007/1 – STORK 1 : D5.7.3 Functional Design for PEPS, MW models and interoperability

[5] K. Kent, M. Souppaya. Guide to Computer Security Log Management. Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-92, September 2006

[6] SANS Consensus Policy Resource Community - Information Logging Standard, http://www.sans.org/security-resources/policies/server-security

[7] NIST: An Introduction to Computer Security: The NIST Handbook, NIST Special Publication 800-12, December 1997, http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf

[8] Common Criteria: Common Criteria for Information Technology Security Evaluation, Version 3.1, revision 4, September.2012 Part 2: Security Functional Components, http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R4.pdf

[9] ENISA: Privacy Features of European eID Card Specification, Version 1.0.1, January 2009, http://www.enisa.europa.eu/doc/pdf/deliverables/enisa_privacy_features_eID.pdf

[10] EC council - The use of audit trails in security systems http://www.europeanpaymentscouncil.eu/index.cfm/knowledge-bank/epc-documents/the-use-of-audit-trails-in-security-systems-guidelines-for-european-banks/

[11] AUDIT Trails - NIST publication http://csrc.nist.gov/publications/nistbul/itl97-03.txt

16. Specific interface

`LightRequest`/`LightResponse` are used for the communication between SP and country-specific implementation, and IdP and country-specific implementation. It will allow countries to use the protocol they want (not only SAML) for the communication between:

- the Specific and the IdP (eIDAS-Node Proxy Service side);
- the Specific and the SP (eIDAS-Node Connector side).

Appendix A.eIDAS-Node Error Codes and Error Messages

The following table provides a list of eIDAS-Node errors and related error messages.

Table 40: Error Codes and Error Messages

Key	Description
sProviderAction.invalidSPQAA.code	Invalid SAML Parameter SP QAA Level error code on the ServiceProvider interface.
sProviderAction.invalidSPQAA.message	Invalid SAML Parameter SP QAA Level error message on the ServiceProvider interface.
sProviderAction.invalidQaaSPid.code	Invalid SP QAA Level error code on the ServiceProvider interface.
sProviderAction.invalidQaaSPid.message	Invalid SP QAA Level error message on the ServiceProvider interface.
sProviderAction.invalidSpId.code	Invalid SP ID error code on the ServiceProvider interface.
sProviderAction.invalidSpId.message	Invalid SP ID error message on the ServiceProvider interface.
domain.ServiceProviderAction.code	Invalid SP domain error code on the ServiceProvider interface.
domain.ServiceProviderAction.message	Invalid SP domain error message on the ServiceProvider interface.
sProviderAction.invalidSPDomain.code	Invalid SP Domain error message on the ServiceProvider interface.
sProviderAction.invalidSPDomain.message	Invalid SP Domain error message on the ServiceProvider interface.
sProviderAction.invalidCountry.code	Invalid Country selected by the Citizen error message on the ServiceProvider interface.
sProviderAction.invalidCountry.message	Invalid SP Domain error message on the ServiceProvider interface.
sProviderAction.spNotAllowed.code	SP not allowed error code on the ServiceProvider interface.
sProviderAction.spNotAllowed.message	SP not allowed error message on the ServiceProvider interface.
sProviderAction.invalidSaml.code	Invalid SP Saml request error code on the ServiceProvider interface.
sProviderAction.invalidSaml.message	Invalid SP Saml request error message on the ServiceProvider interface.
sProviderAction.invalidSPProviderName.code	Invalid SP Provider Name error code on the ServiceProvider interface.
sProviderAction. invalidSPProviderName.message	Invalid SP Provider Name error message on the ServiceProvider interface.
sProviderAction.invalidSPRedirect.code	Invalid SP Redirect URL parameter error code on the ServiceProvider interface.
sProviderAction.invalidSPRedirect.message	Invalid SP Redirect URL parameter error message on the ServiceProvider interface.
sProviderAction.invalidRelayState.code	Invalid SP Relay State parameter error code on the ServiceProvider interface.

Key	Description
sProviderAction.invalidRelayState.message	Invalid SP Relay State parameter error message on the ServiceProvider interface.
sProviderAction.invalidAttr.code	Invalid SP personal attribute list error code on the ServiceProvider interface.
sProviderAction.invalidAttr.message	Invalid SP personal attribute list error message on the ServiceProvider interface.
requests.ServiceProviderAction.code	"Maximum number of requests from SP allowed" error code on the ServiceProvider interface.
requests.ServiceProviderAction.message	"Maximum number of requests from SP allowed" error message on the ServiceProvider interface.
sProviderAction.invalidSPAlias.code	Invalid SP Alias error code on the ServiceProvider interface.
sProviderAction.invalidSPAlias.message	Invalid SP alias error message on the ServiceProvider interface.
sProviderAction.errorCreatingSAML.code	Generating SAML Token error code on the ServiceProvider interface.
sProviderAction.errorCreatingSAML.message	Generating SAML Token error message on the ServiceProvider interface.
internalError.code	Generic error code.
internalError.message	Generic error message.
attrList.code	Invalid attribute list error code on Specific code.
attrList.message	Invalid attribute list error message on Specific code.
missing.sessionId.code	Missing session id error code.
missing.sessionId.message	Missing session id error code.
invalid.sessionId.code	Invalid session id error code.
invalid.sessionId.message	Invalid session id error code.
invalid.session.code	Invalid session error code.
invalid.session.message	Invalid session error code.
callback.code	Invalid eIDAS-Node Proxy Service callback URL error code on CitizenConsent interface.
callback.message	Invalid eIDAS-Node Proxy Service callback URL error message on CitizenConsent interface.
idp.url.code	Invalid IdP redirect URL error code.
idp.url.message	Invalid IdP redirect URL error message.
IdPSAMLResponse.code	Invalid IdP's SAML Response error code.
IdPSAMLResponse.message	Invalid IdP's SAML Response error message.
serviceSAMLResponse.code	Invalid Proxy Service's SAML Response error code.

Key	Description
serviceSAMLResponse.message	Invalid Proxy Service's SAML Response error message.
authenticationFailed.code	Authentication Failed error code on IdPResponse Interface.
authenticationFailed.message	Authentication Failed error message on IdPResponse Interface.
username.code	Authentication Failed error code on specific code.
username.message	Authentication Failed error message on specific code.
invalidAttributeList.code	Invalid Attribute List error code on Specific Code or IdP Response.
invalidAttributeList.message	Invalid Attribute List error message on Specific Code or IdP Response.
invalidAttributeValue.code	Invalid Attribute Value error code on Specific Code.
invalidAttributeValue.message	Invalid Attribute Value error message on Specific Code.
attVerification.mandatory.code	No consent given error code on the CitizenConsent interface.
attVerification.mandatory.message	No consent given to mandatory error message on the CitizenConsent interface.
attValue.mandatory.code	Missing attribute value error code.
attValue.mandatory.message	Missing attribute value error message.
hash.error.code	Hash generation error code.
hash.error.message	Hash generation error message.
qaaLevel.code	Invalid QAALevel error code on Specific Code or IdP Response.
qaaLevel.message	Invalid QAALevel error message on Specific Code or IdP Response.
colleagueRequest.invalidSAML.code	Invalid SAML Token error code on the ColleagueRequest interface.
colleagueRequest.invalidSAML.message	Invalid SAML Token error message on the ColleagueRequest interface.
colleagueRequest.invalidCountry.code	Invalid Citizen Country error code on the ColleagueRequest interface.
colleagueRequest.invalidCountry.message	Invalid Citizen Country error code on the ColleagueRequest interface.
colleagueRequest.errorCreatingSAML.code	Generating SAML Response error code on the ColleagueRequest interface.
colleagueRequest.errorCreatingSAML.message	Generating SAML Response error code on the ColleagueRequest interface.
colleagueRequest.invalidQaa.code	Invalid SP QAA Level error code on the ColleagueRequest interface.
colleagueRequest.invalidQaa.message	Invalid SP QAA Level error message on the ColleagueRequest interface.
colleagueRequest.attrNull.code	Invalid personal attribute list error code on the ColleagueRequest interface.

Key	Description
colleagueRequest.attrNull.message	Invalid personal attribute list error message on the ColleagueRequest interface.
colleagueRequest.invalidRedirect.code	Invalid SP return URL error code on the ColleagueRequest interface.
ColleagueRequest.invalidRedirect.message	Invalid SP return URL error message on the ColleagueRequest interface.
colleagueRequest.invalidDestUrl.code	Invalid Citizen Consent URL error code on the ColleagueRequest interface.
colleagueRequest.invalidDestUrl.message	Invalid Citizen Consent URL error code on the ColleagueRequest interface.
serviceRedirectUrl.code	Invalid Proxy Service Redirect configuration URL error code on the ServiceProvider interface.
serviceRedirectUrl.message	Invalid Proxy Service Redirect configuration URL error message on the ServiceProvider interface
service.attrNull.code	Invalid personal attribute list on eIDAS-Node Proxy Service error code.
service.attrNull.message	Invalid personal attribute list on eIDAS-Node Proxy Service error message.
citizenNoConsent.mandatory.code	No consent given error code on the ConsentType interface.
citizenNoConsent.mandatory.message	No consent given to mandatory error message on the ConsentType interface.
colleagueResponse.invalidSAML.code	Invalid SAML Token error code on the ColleagueResponse interface.
colleagueResponse.invalidSAML.message	Invalid SAML Token error message on the ColleagueResponse interface.
citizenResponse.mandatory.code	No consent given error code on the ConsentType interface.
citizenResponse.mandatory.message	No consent given to mandatory error message on the ConsentType interface.
auRequestIdError.code	The SAML Response Id is either null or not valid error code.
auRequestIdError.message	The SAML Response Id is either null or not valid error message.
audienceRestrictionError.code	SAML Conditions are not complaint error code.
audienceRestrictionError.message	SAML Conditions are not complaint error message.
connectorSAMLResponse.code	Generating eIDAS-Node Connector SAML Response error code on the ColleagueResponse interface.
connectorSAMLResponse.message	Generating eIDAS-Node Connector SAML Response error message on the ColleagueResponse interface.

Appendix B.eIDAS Levels of Assurance

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.

The eIDAS implementing regulation determines three Levels of Assurance:

- **Low** (service.LoA=http://eidas.europa.eu/LoA/**low**)
- **Substantial** (service.LoA=http://eidas.europa.eu/LoA/**substantial**)
- **High** (service.LoA=http://eidas.europa.eu/LoA/**high**)

(The eIDAS-Node Proxy Service `service.LoA` key is described in Table 7.)

At the SAML Request level, the level of assurance will limit the comparison attribute to 'minimum':

- <saml2p:RequestedAuthnContext Comparison="minimum">

Validations made:

At the eIDAS-Node Proxy Service, if the requested (or higher) Level of Assurance cannot be fulfilled, the Request MUST be rejected.

The eIDAS-Node Connector verifies that the Level of Assurance indicated in the Assertion matches or exceeds the requested Level of Assurance, and sends the received authenticated person identification data to the requesting relying party.

The legal definitions of the Level of Assurance can be found at http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=OJ:JOL_2015_235_R_0002&from=EN.

Appendix C. User Consent

In most Member States (MS), the privacy legislation requires that the user gives consent to the use of personal data. But the explanation of this requisite, and thus its implementation may be very different from one MS to another MS. So this general objective to request the consent of the user to send his/her attributes to a Service Provider in another Member State leads to the following consent-schemes. The consent is requested by the eIDAS-Node or by the Middleware of the user's MS.

There are three possible cases:

- The requested attributes are displayed and the user's consent is given by choosing only the attributes that he/she allows to transfer;
- The obtained values of the requested attributes are displayed and the user's consent is given by choosing only the attributes that he/she allows to transfer;
- The requested attributes are not displayed because the user's consent is not required as it was given (for example) when the user registered to the ID Provider.

Appendix D. Hazelcast proposed configuration

To correlate between request/response messages, and to prevent a replay of SAML requests, a caching mechanism is implemented at the eIDAS-Node Connector and Proxy Service level.

For clustered production mode (see section *7 — eIDAS-Node compliance*), the application needs to be configured using Hazelcast product, which will provide a reliable solution based on a distributed hashmap, cluster-ready and with expiration of requests. The configuration of the product is done via its configuration file `hazelcast.xml` located by `EIDAS_CONFIG_REPOSITORY`. A default configuration is provided with the application. It is also possible to implement other clustering solutions by enriching the provided code. Please note, the provided configuration does not cover persistence. If persistence is required, a central database and `MapStore` interface must be implemented. Spring injection of map provider makes it possible on an entry level.

Hazelcast maps are activated by setting `distributedMaps` to "true" in `eidas.xml`.

D.1 Network configuration

The join configuration element is used to enable the Hazelcast instances to form a cluster, i.e. to join the members. Three ways can be used to join the members:

- multicast;
- discovery by TCP/IP;
- discovery by AWS (EC2 auto discovery).

D.1.1 Multicast

In the default configuration, we recommend the multicast configuration for clustering use.

With the multicast auto-discovery mechanism, Hazelcast allows cluster members to find each other using multicast communication. The cluster members do not need to know the concrete addresses of the other members, they just multicast to all the other members for listening. It depends on your environment whether multicast is possible or allowed.

The following is an example declarative configuration.

```
<network>
     <join>
         <multicast enabled="true">
             <multicast-group>224.2.2.3</multicast-group>
             <multicast-port>54327</multicast-port>
             <multicast-time-to-live>32</multicast-time-to-live>
             <multicast-timeout-seconds>2</multicast-timeout-seconds>
             <trusted-interfaces>
                <interface>192.168.1.102</interface>
             </trusted-interfaces>
         </multicast>
```

```
            <tcp-ip enabled="false">
           </tcp-ip>
            <aws enabled="false">
            </aws>
        </join>
   <network>
```

Figure 32: Example Hazelcast multicast declarative configuration

Note: The `multicast-timeout-seconds` element is significant. This specifies the time in seconds that a node should wait for a valid multicast response from another node running in the network before declaring itself as the leader node (the first node joined to the cluster) and creating its own cluster. This only applies to the startup of nodes where no leader has yet been assigned. If you specify a high value to multicast-timeout-seconds, such as 60 seconds, it means that until a leader is selected, each node will wait 60 seconds before moving on. Be careful when providing a high value. Also be careful to not set the value too low, or the nodes may give up too early and create their own cluster.

D.1.2 Discovery by TCP/IP Cluster

If multicast is not preferred as the way of discovery for your environment, then you can configure Hazelcast for full TCP/IP cluster. As the configuration in Figure 33 shows, when the `enable` attribute of multicast is set to `false`, `tcp-ip` has to be set to `true`. For the none-multicast option, all or a subset of nodes' hostnames and/or IP addresses must be listed. Note that not all of the cluster members have to be listed there but at least one of them has to be active in the cluster when a new member joins. The `tcp-ip` tag accepts an attribute called `connection-timeout-seconds` (default value =5). Increasing this value is recommended if you have many IPs listed and members cannot properly build up the cluster.

```
<hazelcast>
    ...
    <network>
        <port auto-increment="true">5701</port>
        <join>
            <multicast enabled="false">
                <multicast-group>224.2.2.3</multicast-group>
                <multicast-port>54327</multicast-port>
            </multicast>
            <tcp-ip enabled="true">
                <member>machine1</member>
                <member>machine2</member>
                <member>machine3:5799</member>
                <member>192.168.1.0-7</member>
                <member>192.168.1.21</member>
            </tcp-ip>
        </join>
        ...
    </network>
    ...
</hazelcast>
```

Figure 33: Example Hazelcast configuration for TCP/IP discovery

D.1.3 Discovery by AWS (EC2 auto discovery)

Hazelcast supports EC2 auto discovery. For information on this configuration please refer to the Hazelcast documentation at http://docs.hazelcast.org/docs/3.2/manual/html-single/.

D.1.4 Eviction

Hazelcast also supports policy based eviction for distributed maps. Currently supported eviction policies are LRU (Least Recently Used) and LFU (Least Frequently Used). This feature enables Hazelcast to be used as a distributed cache. If `time-to-live-seconds` is not 0, entries older than `time-to-live-seconds` value will be evicted, regardless of the eviction policy set. In the application, for anti-replay/reply request-pair correlation cache we set by default the `time-to-live-seconds` to 300 (five minutes) and for the cache of metadata to one day.

```
<hazelcast>
    ...
<map name="antiReplayCacheService">
<time-to-live-seconds>300</time-to-live-seconds> <!-- 5 minutes -->
<eviction-policy>LRU</eviction-policy>
<max-size policy="PER_NODE">500</max-size>
</map>
<map name="antiReplayCacheConnector">
<time-to-live-seconds>300</time-to-live-seconds><!-- 5 minutes -->
<eviction-policy>LRU</eviction-policy>
<max-size policy="PER_NODE">500</max-size>
</map>
<map name="eidasmetadata">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
</hazelcast>
<map name="specificSpRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
<map name="connectorRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
<map name="proxyServiceRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
<map name="specificIdpRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
```

```
<eviction-policy>LRU</eviction-policy>
</map>
<map name="specificConnectorLtRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
<map name="specificServiceLtRequestCorrelationCacheService">
<in-memory-format>BINARY</in-memory-format>
<time-to-live-seconds>86400</time-to-live-seconds><!-- 1 day -->
<eviction-policy>LRU</eviction-policy>
</map>
```

Figure 34: Hazelcast eviction policy configuration

For more information on the features of this product, please refer to the Hazelcast official documentation (http://docs.hazelcast.org/docs/3.2/manual/html-single/).

Appendix E. Installation Frequently Asked Questions

Q: How can I use my SPECIFIC jar?

A: Just replace the `eidas-specific.1.0.jar`, found in the `EidasNode.war`, by your own.

Q: How can I compile the project using external properties (Tomcat)?

A: First you compile EIDAS-NODE and EIDAS-Specific without the "`-P embedded`" argument. This will generate the packages without specific properties. Now you need to place all the properties files in one folder and tell Tomcat to lookup that folder.

If in Linux:

Edit `$TOMCAT_HOME/bin/catalina.sh` and change `"CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/bootstrap.jar"` to `"CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/bootstrap.jar:/path/to/config/folder/"`

If in Windows:

Edit `$TOMCAT_HOME/bin/catalina.bat` and change `"CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/bootstrap.jar"` to `"CLASSPATH="$CLASSPATH""$CATALINA_HOME"/bin/bootstrap.jar:/path/to/config/folder/"`

Q: I'm getting an error that says "Failed to load class org.slf4j.impl.StaticLoggerBinder" .

A: This error is reported when the org.slf4j.impl.StaticLoggerBinder class could not be loaded into memory. In this case, you should recompile your projects to ensure that Maven includes the appropriate jars.

Q: I'm getting an error that says "com.opensymphony.xwork2.DefaultActionInvocation.invokeAction (DefaultActionInvocation.java)" .

A: The `DefaultActionInvocation` class is responsible for calling the user action, if an error occurs, generally due to missing libraries or missing properties file, the struts framework will not be able to render the result of the action, thus producing that error message.

However, in the logs or the stack trace you can usually find another exception. That exception is the reason for this error, perhaps you can solve it by making sure:

- you have the properties files in the right place
- you have the right privileges to access jks file (you may need to install JCE and allow Java to read the file outside the webapp context)
- you have all the required libraries.